A HYBRID ANALYTICAL/SIMULATION MODELING APPROACH FOR PLANNING AND OPTIMIZING MASS TACTICAL AIRBORNE OPERATIONS

by

DAVID DOUGLAS BRIGGS
M.S.B.A., Boston University, 1993
B.S., Northeastern University, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of
Master of Science
College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1995

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
MAY 1995

3. REPORT TYPE AND DATES COVERED
TECHNICAL REPORT THESIS

4. TITLE AND SUBTITLE
A HYBRID ANALYTICAL SIMULATION MODELING APPROACH FOR PLANNING AND OPTIMIZING MASS TACTICAL AIRBORNE OPERATIONS

5. FUNDING NUMBERS

6. AUTHOR(S)

DAVID D. BRIGGS

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

USMA OPERATIONS RESEARCH CENTER
WEST POINT, NEW YORK 10996

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
UNIVERSITY OF CENTRAL FLORIDA, ORLANDO, FL

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
THESIS WORK

12a. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
A hybrid analytical/simulation model is developed to represent and solve the problem of mass tactical airborne operations in an efficient manner. The net result of the hybrid model is an application enabling users to properly load aircraft according to the mission and user specifications. The first portion is a mathematical model of the deployment which generates the impact points of each entity under ideal conditions. This analytical model is represented by a transportation network, and then optimized using a weighted transportation algorithm. The results of this solution are used in an integrated simulation model that introduces the inherent variability. The simulation returns to the user the expected, best, and worst arrival times, and the build up of power over time. The net result of the hybrid model is a tool that allows for effective planning given the available information, as well as simulation results that predict the outcome of the plan. This hybrid approach allows a very large problem to be solved efficiently, and provides analysis of probable outcomes for planning.

14. SUBJECT TERMS

OPTIMIZATION, HYBRID MODELS, AIRBORNE OPERATIONS

15. NUMBER OF PAGES
158

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT

ABSTRACT

A hybrid analytical/simulation model is developed to represent and solve the problem of mass tactical airborne operations in an efficient manner. The net result of this hybrid model is an application enabling users to properly load aircraft according to the mission and user specifications. The first portion is a mathematical model of the deployment which generates the impact points of each entity under ideal conditions. This analytical model is represented by a transportation network, and then optimized using a weighted transportation algorithm. The results of this solution are used in an integrated simulation model that introduces the inherent variability. The simulation returns to the user the expected, best, and worst arrival times, and the build up of power over time. The net result of the hybrid model is a tool that allows for effective planning given the available information, as well as simulation results that predict the outcome of the plan. This hybrid approach allows a very large problem to be solved efficiently, and provides analysis of probable outcomes for planning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

A complex system can be defined as a system with a large number of attributes and often has multiple external influences acting upon it. Also, these large scale problems are not static, and evolve continually over time which makes their representation a difficult task for analysts and operations research model builders. The most difficult of complex problems are those that contain a behavioral element in the decision making process demonstrating the presence of conflict of interest between multiple goals [Verumi, 1978].

Analytical and approximate models have been developed for solving a multitude of complex problems over time. A benefit of mathematical models is that they can be used to find an optimal solution. However, through generalization and simplifying assumptions, these analytical or approximation models may be inadequate for proper representation of a planning system. This becomes especially evident when other than first-come-first-served practices are used in queuing, non-exponential service times are present, machine failures are common, and routing procedures that are possibly Markovian are present. Thus, simulation modeling presents itself as an excellent alternate tool for planning because it allows for the modeling of highly complex systems, but is limited to being descriptive in nature [Shanthikumar, 1981].

One approach for solving complex problems is to isolate portions of the system and model them separately. These models must realistically interact with one another and be able to share requisite data. In order to benefit from the advantages of both analytical and simulation modeling, it may be desirable to combine their methodologies into a hybrid model [Sargent, 1994]. Hybrid modeling is the development and integration of more than one type of representational model, that as a whole can be used to solve or portray a complex problem that would otherwise be too difficult to solve using conventional modeling techniques.

The purpose of this research is to demonstrate how a hybrid analytical/simulation model can be used recursively in order to minimize the disadvantages of mathematical modeling, while gaining new insight through the simulation model [Ignall, 1979]. This approach will be used to solve the complex application of loading paratroopers, by unit, optimally onto aircraft and then determine the expected results once they have arrived onto the ground. The solution to this problem provides the optimal way (in a deterministic sense) to load an aircraft for a particular operation to meet the commander's objectives of rapid assembly towards pre-designated points on the ground.

The problem being addressed in this research is an important problem because inadequate planning for mass tactical airborne operations into combat results in the loss of battle synchronization, poor mission efficiency, and ultimately the loss of human life. While history has recorded paratroopers as *shock troops* for their quick effects on the enemy, unsatisfactory planning of an operation can quickly lead to disaster. Thus, planning for these complex missions is critical for success. The research addressed here is important for the field of industrial engineering because it illustrates the use of two models used in unison to solve a complex problem that would otherwise be inadequately solved with just one type of modeling.

The current and accepted air plan practices for all U.S. Army airborne units is found in the 82nd Airborne Division's Airborne Standard Operating Procedures (ASOP). According to the ASOP, the manifest order should be determined by drawing a schematic of the intended drop zone (DZ), and then drawing a line every 75 yards perpendicular to the line of flight, starting at the personnel point of impact (PPI, where the first jumper will land). Visual estimates are made to determine the closest objective area to each line, as seen in Figure 1. This rough estimation is then used for unit assignment and consequential personnel assignment. The process as specified is laborious, and very rarely carried out in practice.

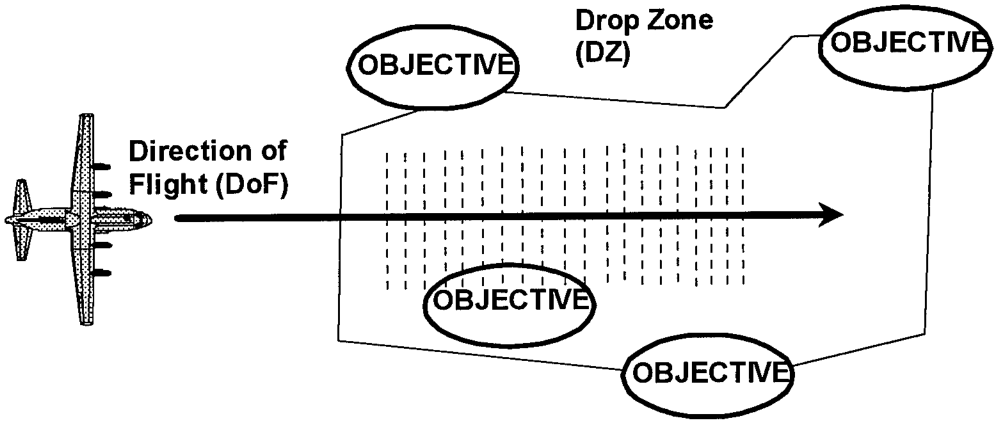


Figure 1. Current Practice Drop Zone Representation

In the operation of a mass tactical airborne mission, the one portion the unit commander has direct impact on is what personnel will load onto an aircraft, and in what order they will be loaded. The seating assignment of the personnel, of course, directly impacts *where* each individual trooper lands. In the ground tactical plan, each trooper (as part of a unit of troopers) is assigned an objective that he must move towards immediately after hitting the ground. Therefore, a good air plan is crucial in the execution of the ground tactical plan.

In order to solve this assignment problem, a purely deterministic model, "*as if the world were perfect*" scenario, will be established and solved to obtain the optimal loading plan. The solution is obtained by minimizing the travel time of paratroopers between the point of impact and their objectives. The solution is then fed into a simulation model that introduces the inherent variability. The sources of variability include aircraft speed, direction and altitude, along with jumper exit intervals, wind drift, descent rate, paratrooper impact recovery, orientation and ground speed to the objectives. The output of this simulation includes the expected mission completion times, maximum and minimum assembly times as well as the variances of the combat power percentages over time for each ground objective.

A hybrid analytical/simulation was selected for this problem because without the variability, the analytical model only provides a solution under optimal conditions (a starting point). A simulation model is necessary because the world is not perfect, and the commander would like to know what he can expect to have happen.

The organization of this thesis is as follows: Chapter two reviews the foundation and characterization of hybrid modeling. Chapter three describes the mathematical representation of the problem and presents the simulation model. Chapter four discusses the computer application and integration of the hybrid model. Chapter five presents the trial results of simulation runs, efficiency and accuracy comparisons to present methods, and some beta version testing of the software package. Finally, chapter six offers conclusions that can be drawn from this research.

CHAPTER TWO
LITERATURE REVIEW

Early Modeling of Complex Systems

In the late 1600's, the German mathematician Liebniz believed that truth was analytical, and that a system could be defined completely by a formal or symbolic procedure. This Leibnizian Approach is the foundation of early system analysis and model building. Leibniz also proposed the methods of decomposition and aggregation for the analysis of large scale systems in order to reduce the dimensionality of the problem at hand [Verumi, 1978].

A large-scale system, as defined by Verumi [1978] is a system in which the number of attributes necessary to describe or characterize the system are too many, the laws relating the properties of the attributes to the behavior of the system are generally statistical in nature, and the system is not static and evolves over time. The behavioral element at the decision making stage also is a major portion of the overall performance of the system.

Decomposition of complex systems is the most straightforward method for analyzing them. Ideally, a complex system can be broken into sub-problems or models and solved in relative isolation. It is the coordination and reintegration of these sub-models that often proves to be the greatest challenge. Problems that are best suited for the Leibnizian approach are ones that have a well defined structure and clearly definable underlying assumptions [Verumi, 1978].

Analytical and simulation models can be considered as two end-points of a spectrum for solving problems. Sometimes it is desirable to combine these modeling approaches in order to solve complex problems. This is referred to as hybrid modeling [Sargent, 1994]. Sargent [1994] defines hybrid modeling as:

> "A ***hybrid model*** is a mathematical model which combines identifiable simulation and analytical models".
>
> ***"Hybrid modeling*** consists of building independent analytical and simulation models of the total system, developing their solution procedures, and using their solution procedures together for problem solving".

Types of Hybrid Modeling and Applications

Ignall, Kolesar and Walker [1978], demonstrated that analytical models are generally preferred by analysts instead of large simulation models. However, in many cases the simplified conditions assumed by the analytical model do not hold true in the real world. A more realistic analytical model often grows so large and unwieldy that it cannot be solved by itself, thus simulation models can be developed to represent the large complex systems.

Ignall [1978] examined the scheduling and routing of fireman and policeman in a large simulation model, and then was able to derive from the simulation runs, a closed form algebraic solution for scheduling. The derived analytical solution method functioned equally as well in developing a schedule when compared to course-of-action alternative analysis used in the simulation model. Refer to Figure 2 which shows the parallel relationship between analytical and simulation models used for validation [Shanthikumar, 1983].

Schwetman [1978] , demonstrated that by combining a discrete-event simulation and mathematical model, he could achieve a better solution than the large scale simulation-only model, at a significantly reduced computational cost. By utilizing

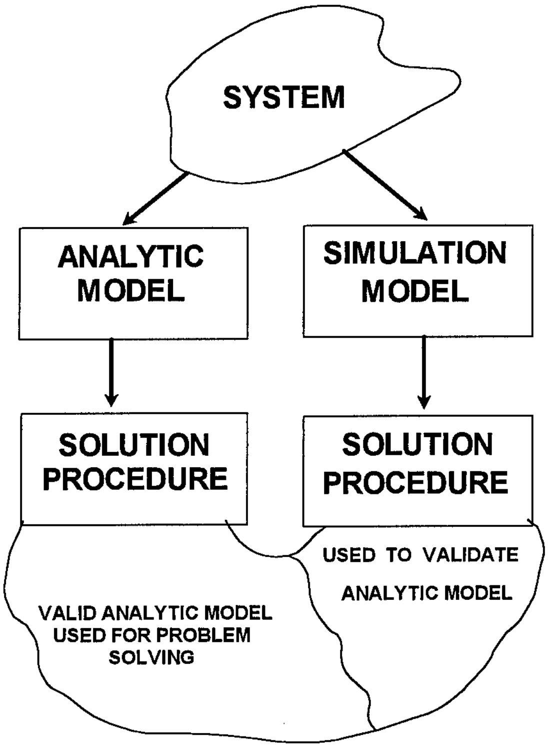

SYSTEM
ANALYTIC MODEL
SIMULATION MODEL
SOLUTION PROCEDURE
SOLUTION PROCEDURE
USED TO VALIDATE ANALYTIC MODEL
VALID ANALYTIC MODEL USED FOR PROBLEM SOLVING

Figure 2. Analytical and Simulation Models Used for Validation [Shanthikumar, 1983]

analytical models in place of portions of the simulation model, he could reduce computational complexity and cost, without sacrificing accuracy. The example used by Schwetman was to divide computer resources into long and short term where the long term requirements were simulated (e.g. arriving jobs, partitioning and establishment of control points). The short term computer resources (e.g. CPU, input and output devices) were modeled using a mathematical model. The net benefit was that the simulation adequately represented those aspects that covered minutes, hours, and days, while the mathematical model handled the tasks that took less than one second. One key benefit in presenting his hybrid model was that it represented the overall system accurately, and appeared to "look" like a realistic computer system. The final benefit of Schwetman's hybrid model was that with a significantly reduced computational requirement, many more scheduling alternatives could be examined.

Shanthikumar and Sargent [1983], and Sargent [1994], present four classes of hybrid models:

1. Class I - Models that alternate over time between independent analytical and simulation models,

2. Class II - Models in which the simulation and analytical model operate in parallel,

3. Class III - Models in which the analytical model is subordinate in some way to the simulation model and,.

4. Class IV - Models in which the overall system is represented by the simulation model, but requires some or all of its inputs from the analytical model.

The following describes these models in detail and presents the application of each class as cited in the literature.

Class I Hybrid Models

A Class I hybrid model is one whose performance over time is derived by alternately using independent analytical and simulation models [Shanthikumar, 1984]. Both models emulate a portion of the overall system, derive a solution procedure independently, and together the independent solutions are brought together for solving the problem. In this approach, both parts of the model function independently. This process is shown in Figure 3.

An example of a Class I type model is a machine shop, with alternating control

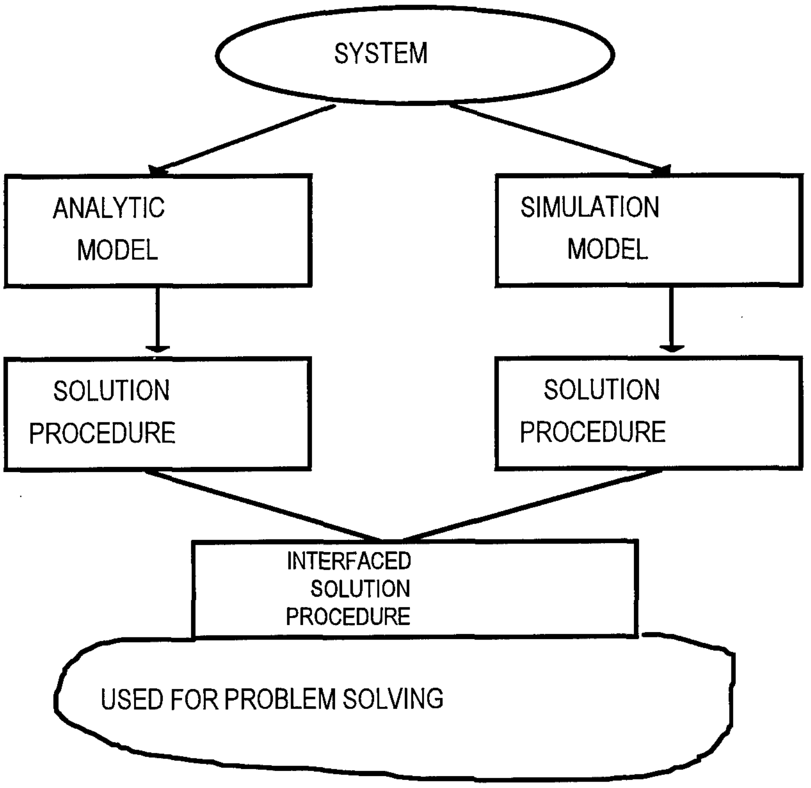


Figure 3. Class I Hybrid Model [Shanthikumar, 1984]

and busy phases. Analytical representations are readily available to model the customer arrival patterns, while the operation or busy time can be more effectively modeled with a simulation model. One model can handle each phase, with the other one starting after the first is complete, and vise versa. The overall methods and times are collected and processed together for problem solving [Shanthikumar, 1984].

Class II Hybrid Models

Class II hybrid models can also be decomposed into an analytical and simulation model. The overall model, however, requires that the two models interact through the solution procedure and to operate in parallel with one another [Sargent, 1994]. Figure 4 shows how each model contributes to the solution procedure and how the two models interact.

In general, the Class II integrated solution method can be much more difficult to implement than the Class I model. Most combined simulation models (having both continuous and discrete portions) can be classified as Class II because the continuous portion often uses a numerical solution method to solve its differential or difference equations, and can therefore be termed the analytical portion [Shanthikumar, 1983]. These two separate models then run continuously, in parallel, with updated solution parameters from one model passing through the solution procedure, updating the other model. As an example, consider a single server queue with a mixed renewal function

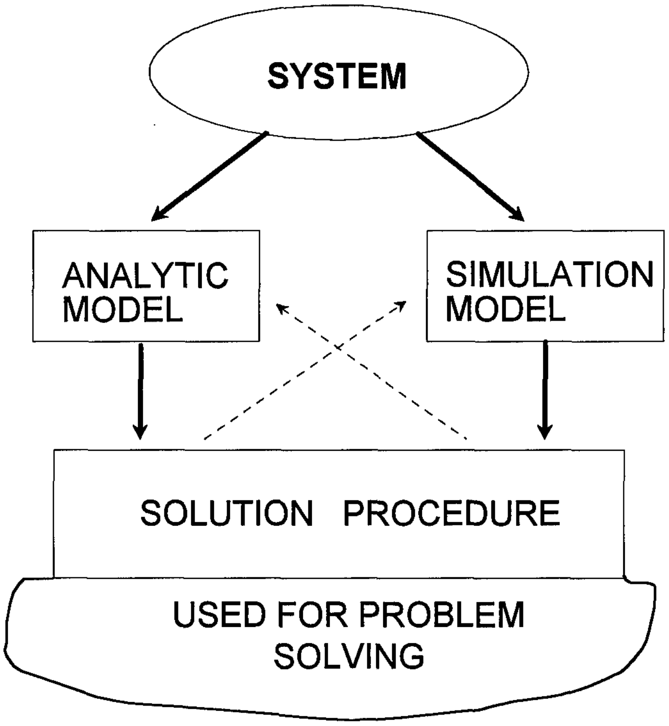


Figure 4. Class II Hybrid Model [Shanthikumar, 1984]

and a Poisson arrival process. The analytical model can derive characteristic sample data that is used by the simulation to form the characteristic server times of the next customer, which then impacts the next customer in the queue coming out of the analytical model, and so on. Each model, in turn, requires data from the other to carry out its processes and to complete its information cycle. These two sub-models are tightly interwoven and are highly dependent [Shanthikumar, 1983].

Class III Hybrid Models

The Class III hybrid model uses both the simulation and analytical model as models of the total system. The simulation model of the total system, or major subsystem, is used to obtain estimates of the input parameters for the analytical model of the entire system [Sargent, 1994]. The analytical model is therefore subordinate to the simulation model, and totally dependent on the outputs of the simulation runs for initialization data, as seen in Figure 5. The analytical model output is then used for problem solving.

An example of a Class III model is a computer CPU queuing system. A simulation model is used to generate the inter-arrival rates and sizes of jobs, which can then be input into as analytical model to optimize the scheduling and execution patterns

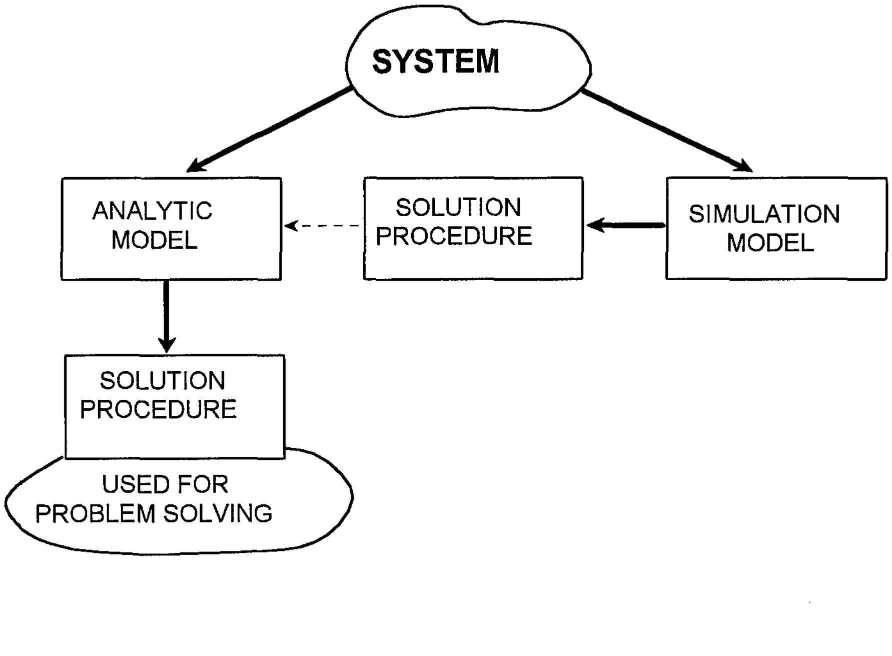


Figure 5. Class III Hybrid Model [Shanthikumar, 1984]

required for the system. Both models contain the functionality of the system, with the simulation model providing inputs for the analytical model [Shanthikumar, 1983].

Class IV Hybrid Models

Class IV models are similar to Class III models, except that the simulation model derives its inputs from the solution procedure of the analytical model [Sargent, 1994]. Class IV models are represented by a simulation model that receives some or all of its input values from the outputs of the analytical model [Sargent, 1994]. In other words, the analytical model provides the initial position or start-up data for the simulation model of the entire system. This process can be seen in Figure 6.

As an example of a Class IV hybrid model, a CPU management system uses an

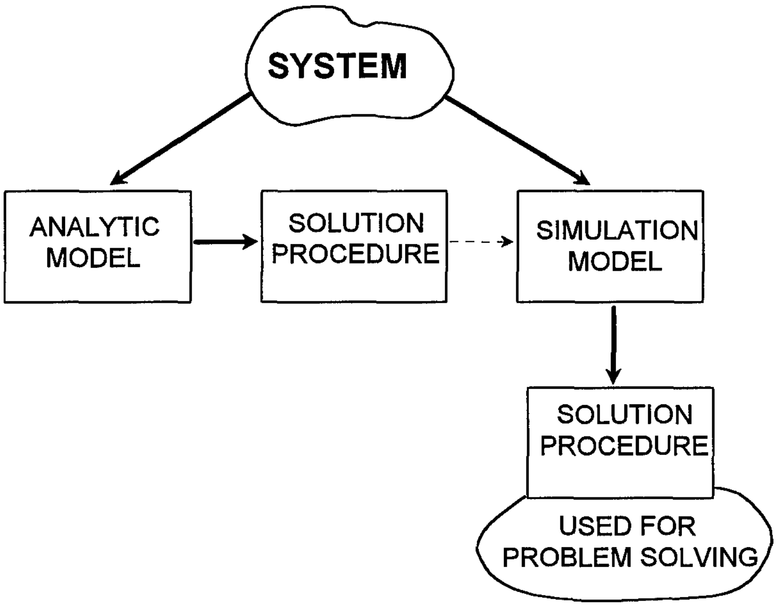


Figure 6. Class IV Hybrid Model [Shanthikumar, 1984]

analytical function to determine the throughput times of scheduled jobs. This solution is then passed to the simulation model which controls the arrival and departure of jobs, and also collects the overall system performance measures [Schwetman, 1978].

Min [1990], demonstrated an adaptation of the Class IV hybrid modeling technique that integrated a large knowledge based hybrid model with flexible manufacturing systems. Min's hybrid model combined a constraint satisfaction problem (analytical model), a rule-based simulation model, and heuristic rules. The technique first generates a solution using the constraint satisfaction problem (CSP) within certain heuristic restraints. The rule-based simulation then evaluates this proposed solution. Finally, a choice rule selects the best solution set and modifies the CSP search space. Intelligent simulation, in this sense, can therefore be classified as a type of hybrid model, with a rule-based, artificial intelligence (AI) or expert system performing instead of, or in addition to, the analytical model along with the simulation model.

Regardless of the type or class, hybrid modeling has been shown to be useful in gaining insight into system behavior, validating analytical models and solving large scale complex problems [Ignall and Kolesar, 1979, Hanssman, 1980]. The approach applies fundamental OR/MS theory by decomposing the large problem into functionalities, and then applying the best representation/solution tool to solve it.

In this research, a Class IV hybrid model has been selected for representing and solving the complex problem of airborne manifesting. Chapter 3 presents the methodology that combines the analytical and simulation models for solving this problem.

CHAPTER THREE
DEVELOPMENT OF THE MODEL

In this research, a Class IV hybrid analytical/simulation model will be used to solve the problem of airborne manifesting. The goal is to determine the best way to load the aircraft, according to the commander's objectives and intent on the ground. The commander's objective is to minimize the required travel time for each jumper from his point of impact to his given assembly point, in order to support the accomplishment of the overall mission. The first component of the model is a mathematical model that provides an optimal loading strategy by minimizing the total travel time for a given operation. This model will be solved with the assumption that everything occurs exactly as planned. To better reflect reality, the output of the analytical model (i.e. the optimal loading strategy) is input into a discrete event simulation, which is the second component of the model, in order to give commanders feedback on what they may expect to happen on the ground. The simulation model introduces the inherent variability of airborne operations such as wind conditions, drift, aircraft location and speed, delays between jumper exits, and other instances of variance. The simulation allows for the comparison, validation and verification of portions of the analytical model. Figure 7 demonstrates the proposed approach.

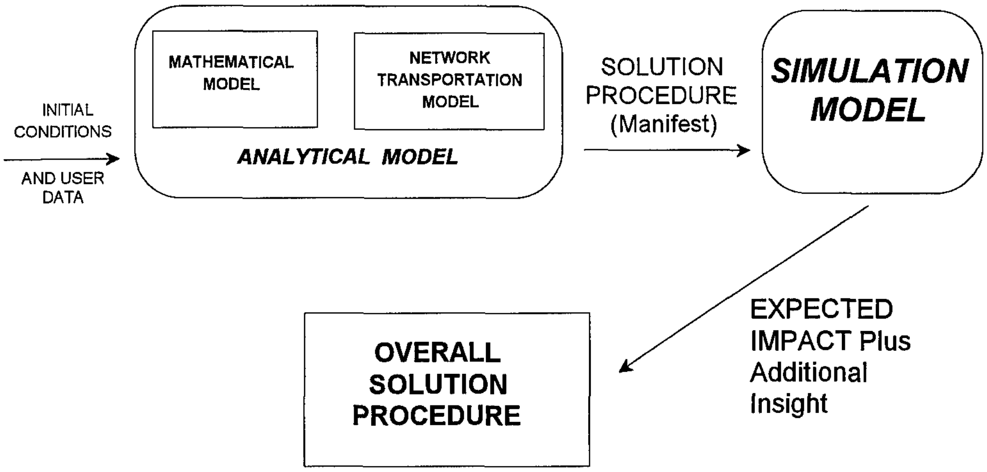

INITIAL CONDITIONS
AND USER DATA
MATHEMATICAL MODEL
NETWORK TRANSPORTATION MODEL
ANALYTICAL MODEL
SOLUTION PROCEDURE (Manifest)
SIMULATION MODEL
EXPECTED IMPACT Plus Additional Insight
OVERALL SOLUTION PROCEDURE

Figure 7. Proposed Class IV Hybrid Model Approach

Mathematical Model

The mathematical model represents the distributions and dispersal of jumpers out of each aircraft, their eventual position on the ground and the relative distance to each objective. In order to formulate this model, a collection of initial assumptions are made which is the basis for the *perfect world* model:

1. All aircraft are traveling at constant 125 knots per hour.
2. All aircraft are flying along the exact heading as specified.
3. The flight formation is in-line trail, which means each aircraft in a serial follows directly behind the aircraft ahead, with no displacement left or right.
4. Each aircraft begins dropping jumpers at the Calculated Air Release Point (CARP) and that the CARP was correctly computed to have the first jumper

from the primary jump door impact directly on the Personnel Point of Impact (PPI).

5. The interval time between aircraft in a serial of aircraft is minimal (usually less than 10 seconds).
6. Time under the parachute canopy is equivalent for all jumpers and is minimal.
7. Ground elevation is uniform (i.e. the world is flat).
8. Jumpers exit uniformly at one second intervals from each door.
9. The assistant jumpmaster's door begins exiting jumpers exactly one-half second after the primary door begins.
10. All aircraft are cross-loaded equally (i.e. units are equally dispersed over all of the available aircraft in order to minimize risk if an aircraft is lost).
11. Distances are calculated on a straight line basis of travel.

In order to calculate where each jumper should impact on the ground, the Military Grid Reference System (MGRS) was used to determine location. The *x*-coordinate is a false Easting value expressed in meters, increasing from left to right. The *y*-coordinate is the Northing component, and increases from bottom to top. A ten digit system is used (5 for each *x* and *y*) that records the locations accurately to one meter (see Figure 8). Angles are measured from grid north and increase clockwise. All measurements are in meters. This procedure creates a synthetic 2-dimensional *xy* plane to be used in both the

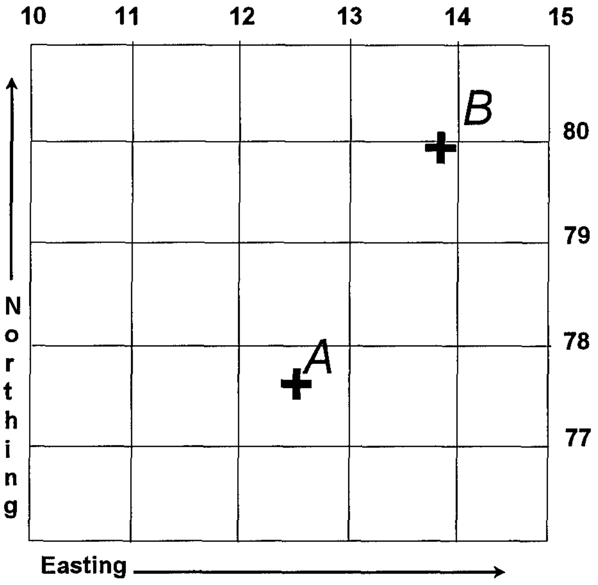


Figure 8. Military Grid Reference System (MGRS)
Through interpolation, point A is located at grid 12500 77500, point B is at 13850 79900.

deterministic and simulation model.

The first jumper exiting from the primary door impacts on the Personnel Point of Impact (PPI) and gains the PPI's *x* and *y* coordinates. Each subsequent jumper's *x* and *y* position must then be calculated incorporating the heading, or Direction of Flight. The following is the formula for determining *x* and *y* coordinates for jumpers one though *n* on the primary door:

$$X_n = X_{n-1} + [\sin(DoF) * \ 68.58 \text{ meters}]$$
$$Y_n = Y_{n-1} + [\cos(DoF) * \ 68.58 \text{ meters}]$$

where:
$Xn=$ *the Easting coordinate for jumper n*
$Yn=$ *the Northing coordinate for jumper n*
$DoF=$: *the direction of flight in degrees from true north*
68.58 meters is equal to 75 yards which is the distance traveled in one second at 125 knots.

The first jumper from the non-primary door ideally exits one half second after the primary door. His location is then found using the formulas below, with jumpers 2 through *n* using the formula above and substituting $X_1$and Y_1's location for X_{n-1} and Y_{n-1}.

$$X_1 = X_{PPI} + [\sin(DoF) * 34.29 \text{ meters}]$$
$$Y_1 = Y_{PPI} + [\cos(DoF) * \ 34.29 \text{ meters}]$$

where:
34.29 meters is the distance traveled in 1/2 second at 125 knots per hour

Once the impact location for each jumper is determined on the ground, the distances to each possible objective can be calculated from each impact point. This is done simply by using Pythagorean's Theorem as follows:

$$Distance = \sqrt{[(X_{OBJ}} - X_n)^2 + (Y_{OBJ} - Y_n)^2]$$

where: X_{OBJ} *and* Y_{OBJ} *are the x and y location of each objective and* X_n *and* Y_n *are the impact locations for each jumper.*

Once all the distances from the impact points to the corresponding objective locations have been determined, they are converted to time factors using the maximum sustainable speeds from the heat stress model found in the Dismounted Infantry Movement Rate Study [Hayes, 1994].

Note that in a given mission, several objective locations exist that the paratrooper must head toward upon impact. These locations are, of course, not of the same importance level. The objectives are generally classified into three categories:

1. Mission Essential - Those missions that absolutely must be accomplished for the overall success of the operation.
2. Mission Support - Units that directly support the execution of the mission essential force.
3. Secondary Mission Support - Those units assigned missions that are not critical to the essential or primary support missions.

The relative importance of each objective was assessed using a pair-wise comparison of the criteria, by the command and staff of the 2nd Battalion, 325th Airborne Infantry Regiment, who served as the client for this project. The relative importance of these objectives were determined to be 5.2, 2.3, and 1.0 for mission essential, mission support, and secondary mission support respectively. These factors will later be used in the formulation of the mathematical model.

Several optimization methodologies were considered in finding an optimal solution for this problem. If one views the original virtually unbounded problem of minimizing the overall travel distances from origin points that displayed wide random placement, it can be readily seen that this problem can be NP (Non-Polynomial) complete. To solve NP complete problems, such as a decision problem, a polynomial transformation must be done in order to change the problem into a polynomial form.

Application of heuristics also can be used to down scale the size of this type of problem. [Evans, 1992].

This problem easily lends itself to network representation, and can be viewed as a network assignment problem utilizing integer values (personnel, aircraft and objectives). In order to determine the optimum manifest of the aircraft, the problem is modeled and solved as a transportation network assignment problem.

The transportation (or distribution) problem pre-dates most linear programming as an optimization method, and is a standard application for industry having multiple sources, destinations, and capacitated routes. In the standard interpretation of the model, there are *m* supply points (or origin nodes) and *n* demand nodes. From the supply nodes, there are a specified number of items needing transport. At the demand nodes there are a specified number of requirements to be met. Arcs are drawn and costs computed from each supply node to each and every possible demand node in the system. The overall formulation for this network minimization problem is as follows [Wagner, 1975].

$$\text{Minimize } \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{subject to: } \sum_{j=1}^{n} x_{ij} \le S_i \text{ for } i = 1, 2, 3 \ldots., m \text{ (supply)}$$

$$\sum_{i=1}^{n} x_{ij} \ge D_j \text{ for } j = 1, 2, 3 \ldots., n \text{ (demand)}$$

$$x_{ij} \ge 0 \text{ for all } i \text{ and } j$$

where:
C_{ij} : *the arc transportation cost from node i to node j*
X_{ij} : *the amount of material transported down from node i to node j.*
S_i : *the supply values at supply node i*
D_j : *the demand values at demand node j*
m: number of supply nodes
n: number of demand nodes

Because the basic feasible solution must be composed of an integer solution, and all manipulation in the transportation tableau is done through addition and subtraction, the resulting solution will be integers. This transportation model not only avoids the inherent complexity of integer programming, but offers a dynamic solution process that has a level of complexity less than that of linear integer programming [Armacost, 1991]. Dummy objective nodes are created in those instances where the quantity of the supply nodes (i.e. the total number of aircraft in the serial) exceed the number of troopers allocated to the designated objectives. For the application problem in this research, these dummy slots will be used for parachute fillers, effectively placing non-essential personnel in positions that are not required by jumpers with mission essential or support missions to be accomplished for the commander.

To formulate the manifest plan as a transportation problem, the seating capacity of the aircraft on both the left and right door are used as an array of source nodes, with the supply value at each node equal to the total number of aircraft (or chalks) in the serial. The demand nodes are the different objectives or assembly areas, with the demand value equal to the number of personnel in the subordinate unit assigned to seize that objective, times negative one (demand nodes are always negative, denoting a need to be filled). The time-distance coefficients are the costs of the arcs connecting the supply nodes (jumper position) with the demand nodes (different objectives). These costs are the weighted travel times computed between each jumper location and each objective. This network is shown in Figure 9.

The network nodes and arcs reduce the total number of variables. For an example with 120 jump positions and 6 objectives, using integer programming, 720 variables would be required, with a majority of them being zero. In the corresponding transportation model the requirements of 126 nodes must be satisfied through the

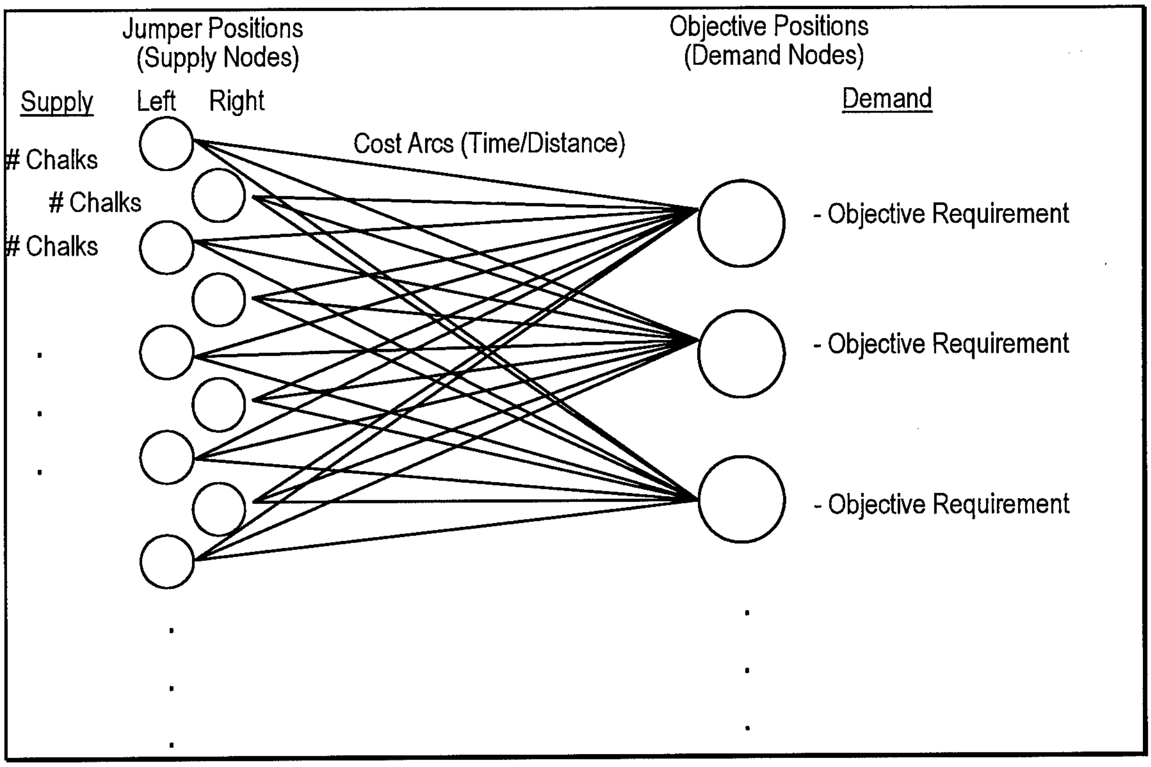


Figure 9. Transportation Network

selection and utilization of the 720 possible capacitated arcs. While costs and penalties must be computed for the usage of each arc, all are not used.

For software implementation, the transportation method is coded using Microsoft Visual Basic for Windows. The main reason for using this software is that once compiled, the application is free-standing and royalty free. See Chapter 4 and Appendix A for the Visual Basic code.

In order to formulate and solve the network, the program constructs an *m* by *n* cost matrix where *m* is the number of total jumpers for one aircraft, and *n* is the number of objectives. An identically sized array of integer values is constructed to represent the solution values. Constraints are placed on the number of jumpers assigned to each objective as well as the number of chalks. The transportation algorithm continues until all constraints have been met, and no further improvements can be made by having non-basic variables enter (i.e. optimality has been achieved).

There are several approaches for generating an initial basic feasible solution for the transportation problem. The methods compared in this research include the North West Corner Rule, Vogel's Approximation Method, and the Minimum Cost Method. The North West Corner Rule is a greedy algorithm that begins in the upper left hand corner of a tableau, allocates as much as possible to each cell or arc, and then moves right and down sequentially filling as much as it can, yielding the required $m + n - 1$ basic variables (m = columns, n = rows). The Minimum Cost Method finds the overall smallest coefficient value, allocates as much as possible to that cell, and then searches for the next smallest coefficient, eliminating rows and columns as they are satisfied for supply and demand respectively.

The Vogel Approximation Method (VAM) operates by computing row and column penalties, finding the largest for the coefficients (i.e. with the highest cost for non-selection), and selects the smallest value in that row or column for allocation. The algorithm then eliminates the satisfied row or column, recomputes row and column penalties, and starts again.

Among the three approaches for finding the initial feasible solution, VAM is the most complex [Winston, 1991]. However, VAM yields an initial basic solution that is very close to the optimum and often yields the optimum solution. Because of the proximity of the initial feasible solution to the optimal solution, VAM requires the fewest iterations of the transportation simplex to find the optimum. The North West Corner Rule on the other hand, requires the least time to compute the initial solution, but takes the most iterations of the simplex to reach optimality. The North West Corner, on the average, takes three times as many pivots as VAM for the same size of problem [Glover et al., 1974]. The Minimum Cost Method falls between the previous two methods in overall complexity. Vogel's Approximation Method has been selected for finding the initial feasible solution in this research.

Simulation Model

To introduce the inherent variability that occurs in airborne operations, the optimal solution obtained from the analytical model is fed into a simulation model in order to give the commander a realistic range of expectation for actions on the ground. The range of variability is examined in terms of actions in the aircraft, actions in the air (after exiting the aircraft) and actions on the ground. This introduction of variability directly addresses the assumptions made in order to formulate the optimization model.

One source of variability comes from the fact that the aircraft may not be flying the exact heading as indicated in the pre-flight briefing. Experience has shown that the plane can deviate approximately 5 degrees in either direction and aircraft speed can range from 110 knots to 130 knots, with a mean of 125. The calculation of the CARP is done by offsetting the estimated throw (the approximately 210 yards that a jumper is thrown forward upon exit prior to canopy envelopment) back against the direction of flight. This figure is further offset 200 yards against the wind direction for each 5 knots of wind, in order to compensate for drift. The CARP is therefore a relatively rough estimate of where troopers should begin to jump so that they will land on the PPI. It is the job of the navigator in each aircraft to announce the CARP so that the co-pilot can turn on the green light to start the jump. Note that a delay of one second at 125 knots, translates to 75 yards (or 68.58 meters).

Once the green light is lit, the number one jumper on the primary door must exit immediately, with each subsequent jumper ideally following the previous jumper at a uniform one second interval. The assistant jumpmaster's door should begin exiting jumpers approximately one-half second after the primary door begins. This jump interval can be perceived as an inter-arrival rate. The actual data and distributions used by the simulation model are from the last round of C-141 testing done in October 1994 by the TEXCOM Airborne and Special Operations Test Directorate.

The actual drop altitude also plays an important role in the amount of inherent drift of the parachute. For every 100 feet above the ground under an inflated canopy, each parachute can drift 4.2 yards in any direction. This drift can be accentuated if the lead aircraft in a serial is flying above or below the requisite 800 ft Above Ground Level (AGL). Trail aircraft in a serial also tend to fly an average of 50 feet above the aircraft in front to avoid hitting paratroopers in the sky. The actual historical data of deviations is not required to be maintained by the U.S. Air Force's Air Mobility Command, thus variances were obtained from area experts' best estimates, and conversations with six navigator/pilots. A portion of the jumpers may also be injured, either in the air or upon impact. However, because this data must be based solely on the estimates of a user, under the given circumstances, it was determined to be an non-influential factor for this model.

Once a jumper impacts the ground, he must immediately get his weapon into operation, roll up his parachute and air items, and begin moving toward his objective.

The approximate time for each trooper to accomplish this maneuver is 7.5 minutes, with a standard deviation of 2.0 minutes. A portion of these jumpers, approximately 5%, will be lost enroute to their objective anywhere from 10 to 60 minutes. On the way to his objective the trooper will travel at the maximum sustainable rate given the temperature, humidity, available light, surface and grade conditions (as defined in Dismounted Infantry Movement Rate Study [Hayes, 1994]).

The inputs for the simulation model include the manifest listing by subordinate unit designation of each aircraft in the serial generated from the mathematical model, using the transportation problem algorithm. Other inputs of the simulation model include starting location, number and type of aircraft, aircraft heading and environmental conditions. The outputs of the simulation are the build-up of combat power over time at each of the objective areas. Critical percentages achieved (e.g. 50%, 75%, 90%, etc.) are tagged and their corresponding code-words generated during each replication. Minimum, maximum, and mean distances and arrival times will be recorded for the commander's review.

Both the optimization and simulation application are discussed in more specific detail in Chapter 4.

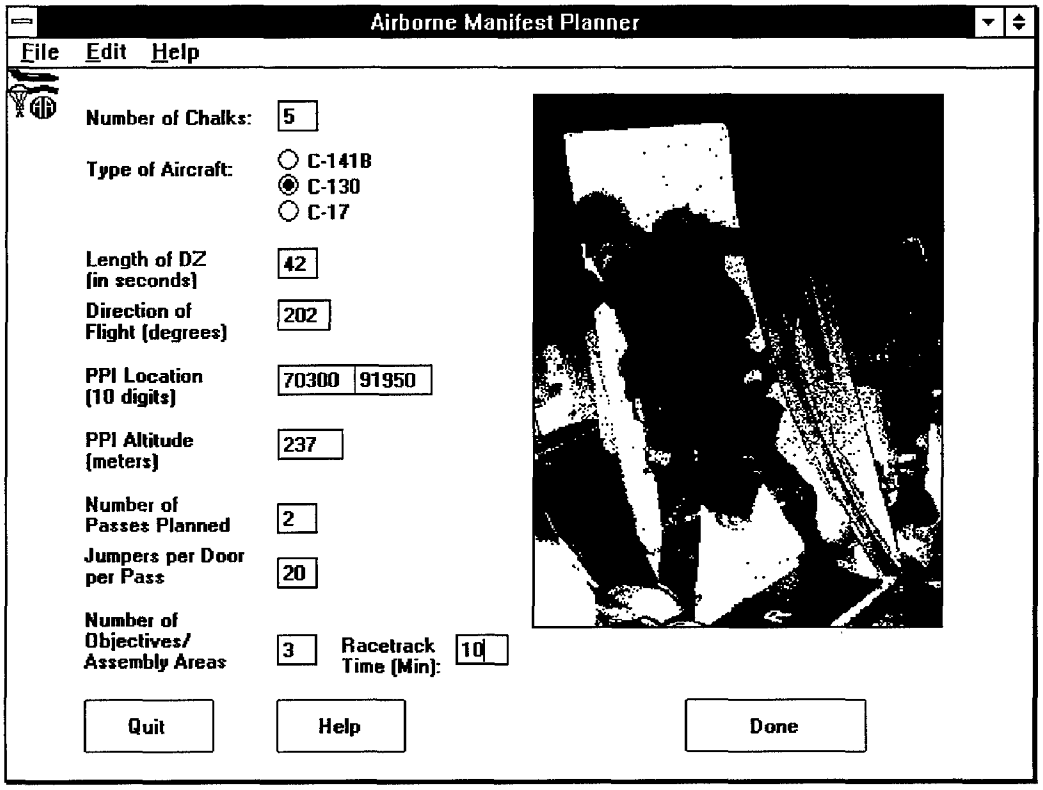
CHAPTER 4
APPLICATION OF HYBRID MODELING TO MASS TACTICAL AIRBORNE OPERATIONS

A Class IV hybrid analytical/simulation modeling approach has been selected to solve the problem of manifesting for mass tactical airborne operations. In order to solve this problem, a mathematical model of the system was first developed and optimized using a transportation network. The results of the optimization are then interfaced with a object-oriented simulation model that introduces variability and allows for additional experimentation. The outputs of the hybrid application are the optimum aircraft manifest, given the commander's requirements, and what the commander can expect on the ground as a result of his plan.

Analytical Model

The overall analytical model is coded in Microsoft's Visual Basic 3.0 for Windows. Visual Basic was selected as the shell because of the readily useable graphical user interface (GUI). Windows is already widely used by the target client, the U.S. Army, and it has the ability to launch and communicate with other Windows applications. Finally, a Visual Basic application is stand alone, generates its own Windows setup disks, and is essentially free to the client for distribution and use.

In order to setup the mathematical model, the values of several input variables must be entered by the user. Figure 10 illustrates the first window that the user encounters in order to provide the initial parameter values. The names and definitions of these initialization parameters are shown in Table 1.



Number of Chalks: 5

Type of Aircraft: ○ C-141B ● C-130 ○ C-17

Length of DZ (in seconds) 42

Direction of Flight (degrees) 202

PPI Location (10 digits) 70300 91950

PPI Altitude (meters) 237

Number of Passes Planned 2

Jumpers per Door per Pass 20

Number of Objectives/ Assembly Areas 3

Racetrack Time (Min): 10

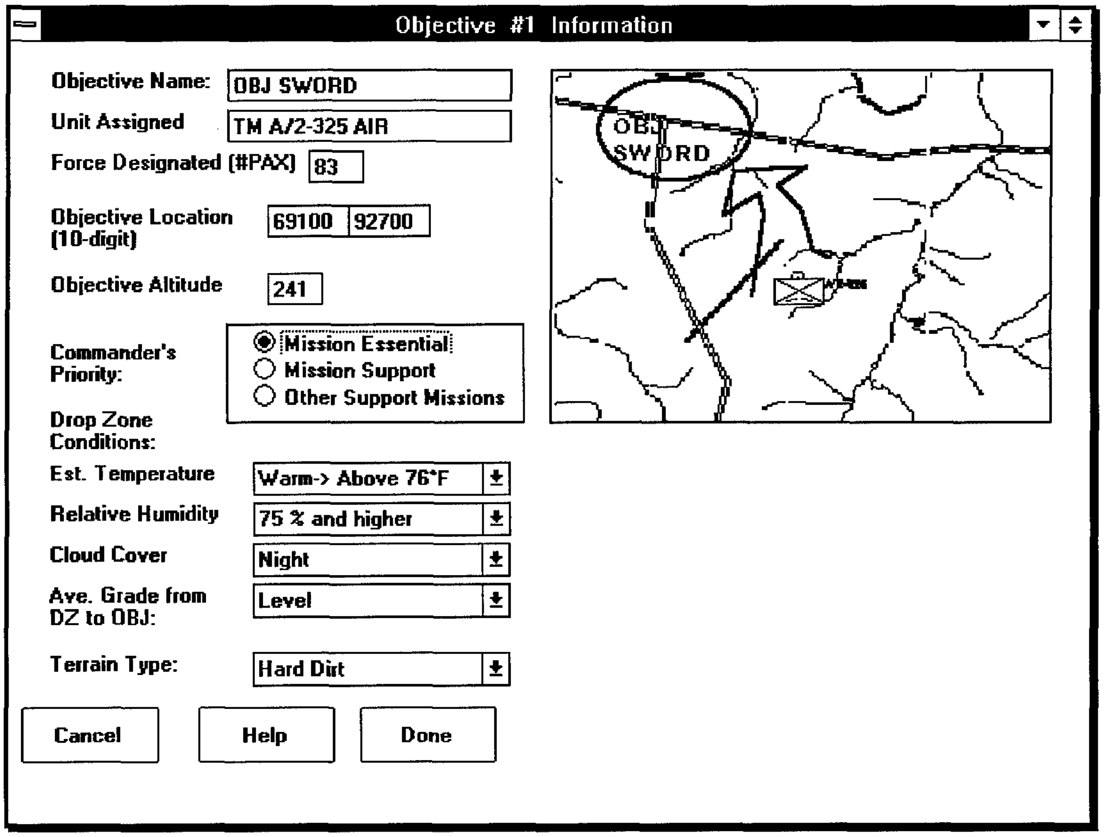
Figure 10. Initialization Parameters Screen

Table 1. Global and Initialization Parameter Values

Parameter Name	Data Type	Definition
Number of Chalks	Integer	Number of aircraft in a serial
Length of DZ (in sec.)	Real	Time length of useable drop zone
Direction of Flight	Real (Degrees)	Direction vector of the aircraft
PPI Location	Real/Real	Expected first impact point location
Number of Passes	Integer	How many times the aircraft will over-fly and drop jumpers
Jumpers per Door	Integer	Number of exits on a pass
Number of Objectives	Integer	Number of demand nodes
Racetrack Time	Real	Amount of time it takes to line up the aircraft and begin the next pass
X Location Array	Integers	The Northing location values, array sized 1..Jumpers per Door
Y Location Array	Integers	The Easting location values, array sized 1..Jumpers per Door

From the initialization parameters, the global arrays and matrices are formed using the following procedures. First the personnel point of impact (PPI) location is used as a starting point, and the distributions of impact points along the heading are computed using the continually updating the position equations described in Chapter 3. These impact points Easting and Northing location values, which are numbered from one to the number of jumpers in a pass, are stored in two X and Y arrays for both sides of the aircraft. Together these arrays are the projected components of ground impact locations of each jumper or supply nodes. Requirements for number and length of passes will be used once the distances are converted to travel time. The user is then queried as to the total number of jumpers in each aircraft (i.e. whether it is full to its type-capacity, or not). The number of designated objectives, sets up the next portion of the problem, displaying a number of screens that request information on each objective.

Each objective is characterized by a unique name, an assigned unit, a defined location, and the number of paratroopers in that unit that are given a mission to seize. Secondly, in order to adequately weigh the overall importance of each objective, a commander's assessment is made of mission priority and type. A sample input screen is displayed in Figure 11. Finally, the user must specify the environmental conditions under which the mission is to be carried out. These conditions are shown in Table 2, and



Objective #1 Information

Objective Name: OBJ SWORD

Unit Assigned: TM A/2-325 AIR

Force Designated (#PAX): 83

Objective Location (10-digit): 69100 92700

Objective Altitude: 241

Commander's Priority:
● Mission Essential
○ Mission Support
○ Other Support Missions

Drop Zone Conditions:

Est. Temperature: Warm-> Above 76°F

Relative Humidity: 75 % and higher

Cloud Cover: Night

Ave. Grade from DZ to OBJ: Level

Terrain Type: Hard Dirt

Figure 11. Objective Information Input Screen

Table 2. Ground Speed Determination Criteria

Parameter	Possible Value Ranges
Ambient Temperature	Cool, below 50 º F Mild, 50 º to 75 º F Warm, greater than 75 º F
Relative Humidity	25% or less 26-74% 75% or more
Cloud Cover	Night Clear Sky Partly Cloudy Cloudy
Average Grade from DZ to OBJ	Level (+/- < 10%) Slopes Up (>10%) Slopes Down (>10%)
Type of Terrain	Asphalt Hard Dirt Loose Sand

contained in a fixed five-dimensional array as a look-up table within the application itself. These environmental factors return a maximum sustainable speed, in meters per second, and are obtained from the Dismounted Infantry Movement Rate Study [Hayes, 1994], that models core temperature and human performance under stress. The entire database of movement speeds was much too large for practical use (18,900 entries), so a representational portion of the significant forces are used in this model, with 324 possible combinations of the five parameters available. A factor not used in this model is risk level. Due to the nature of these type of operations, a high level of risk was assumed throughout.

Once the user enters the required information, the model utilizes the data to calculate the distances from each of the impact point's x and y locations from the global array to the x and y location of each objective, and enters the values into a distance array for that objective column. The distance coefficient array is sized by two times the number of jumpers in a pass, by the number of objectives.

Each of the other objectives' data is then input in sequence using identical objective information screens, with the temperature, humidity and cloud cover parameters remaining fixed, having already been determined in the first objective screen.

After all of the objective data has been entered, and the distance array for the first pass is formed, the distance coefficients must be weighted and converted to time. The coefficients are multiplied by the mission criticality weighting factor, 5.3 for Mission Essential and 2.3 for Mission Support, and then divided by the computed ground speed in meters per second. This returns a weighted travel time in seconds, from each impact point to each objective, and is stored in an array sized by two times the number of jumpers per pass times the number of passes, by the number of objectives.

Time is used as the coefficient unit for the network because additional passes, and the requisite time to complete each flight back around to the drop zone (i.e. racetrack), must be considered in order to adequately evaluate the consequences. This allows the weighing of alternatives between distances (now time) and the additional racetrack time. As an example, the tradeoff can be made by the algorithm to either drop a jumper on an earlier pass farther from his objective, or on a later pass closer to that objective.

Once the first pass time coefficients are computed, the array expands to add the additional passes, adding the racetrack time (in seconds) to each additional pass's weighted travel times. Because the last pass of most operations is uneven, the overall size of the array is reduced to reflect that capacity, by eliminating those positions not required in the last pass. A set of duplicate sized, two dimensional, blank array of integers are formed that serves as the variable matrix for the solution procedure. The following transportation problem equation and constraints are utilized:

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{subject to:} \quad \sum_{j=1}^{n} x_{ij} \leq S_i \ \text{ for } \ i = 1, 2, 3 \ldots., m$$

$$\sum_{i=1}^{n} x_{ij} = D_j \ \text{ for } j = 1, 2, 3 \ldots., n$$

$$x_{ij} \geq 0 \text{ for all } i \text{ and } j$$

where:

C_{ij} : *the weighted travel time coefficients*

X_{ij} : *the number of jumpers that will utilize that ij path.*

S_i : *the number of chalks in this mission*

D_j : *the requirements at each objective*

m: number of jumpers in a single aircraft

n: number of objectives

Next the algorithm constraints must be considered. Each row in the variable array must sum to the number of chalks in the mission (i.e. the quantity in the supply nodes), and each column (objective) must sum to the troop requirement at that demand node. These constraints are stored in two separate, one dimensional constraint arrays, with a duplicate array that functions as the working sum, being less than or equal to the constraints as the solution is developed. The row sum array is one dimensional and sized one to the number of objectives plus one, the column array is also one dimensional and sized one to the total number of jumpers in one aircraft. Also added here is a dummy column (or demand node), that serves to take up the slack between the total force requirements of each column and the total capacity of the aircraft. The coefficients of the dummy column are zeroes, as to not affect the optimal solution, and allow the placement of excess, non-essential personnel into a position not required more by a mission oriented jumper. Now that the transportation network has been established dynamically by the user tool, according to the parameters of the user, the solution procedure can begin. First an initial basic feasible solution must be found.

Transportation Algorithm

Once a basic feasible solution has been determined (using Vogel's Approximation Method) and stored in a basic feasible solution (BFS) array, the search for the optimal solution begins. This is done by utilizing the MODI method. u_i and v_j multipliers are computed during each iteration of the algorithm for each row and column respectively. These multipliers are computed such that:

$u_i + v_j = c_{ij}$, for each basic variable x_{ij}

where:

i and j represent row and column position

u_i = the row penalty value

v_j = the column penalty values

C_{ij} = the weighted travel time coefficients for that cell

The first *u* row multiplier is assigned a value of zero (arbitrarily), and then all subsequent values can be computed by subtracting its compliment from its weighted time coefficient *c*. Each non basic variable must then be assigned a benefit cost for inclusion into the basic matrix (array). These values are determined by tabulating:

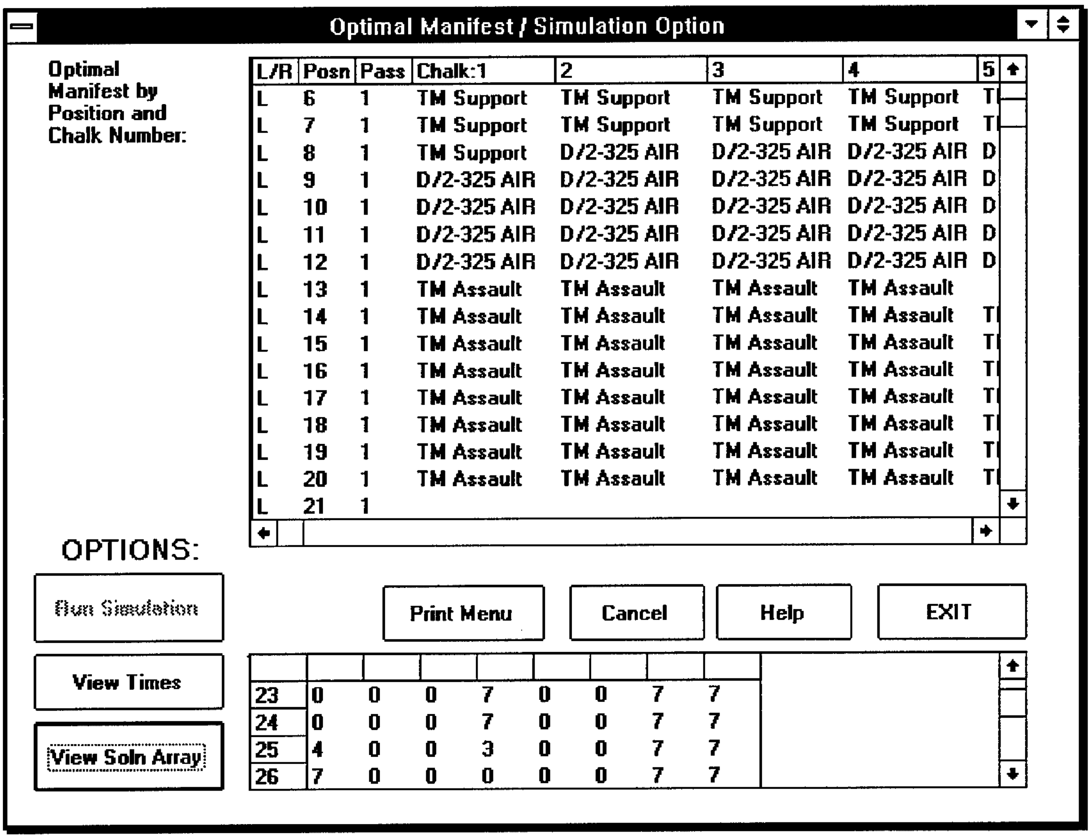
$\bar{c}_{pq} = u_p + v_q - c_{pq}$, for each non basic variable x_{pq}

with p and q representing row and column position, respectively

The cell with the largest, non negative $\bar{c}_{pq}$ value is selected for inclusion into the basic. To find the leaving basic variable, a search is conducted up and down, and on both sides of the entering variable to find the largest possible level of improvement. From this, the smallest variable value is selected, and the corresponding adjacent values adjusted creating a closed loop. This loop allows a value to be changed, and then the ramifications of that new value are re-computed in all cells that were affected. The algorithm then begins again by recomputing the row and column penalties, and continues until there are no longer any positive $\bar{c}_{pq}$ values [Taha, 1992]. The working solution array is then copied into a final solution array, that is the optimal solution.

Once the optimal solution is reached, the next step is to present the solution to the user in an understandable manner. Note that this solution is a manifest listing of where jumpers are to be assigned. This is done by searching the final tableau array row by row, with the left door listing coming from the odd numbered rows, and the right door from the even rows. Each row is searched for non-zero values. If the variable is equal to the number of chalks, each aircraft is assigned a jumper belonging to that objective's column designation. For variables less than the chalk number, the aircraft up to that value are assigned, and then the remaining non-zero basic variables are assigned in that row for aircraft beyond the last number, up until the number of chalks is satisfied for that position. Null values are assigned a unit designation string of a space-key character, and are employed by the user for non-essential personnel fillers. All these transformed values, in the form of unit strings, are placed into a grid for display and review by the user in the application, and can also be printed out for preparation of the actual manifests. See Figure 12 for a sample final "optimum" manifest screen.

This screen also serves as the interface for launching the simulation model, which is the second portion of the hybrid model.



Optimal Manifest by Position and Chalk Number:

L/R	Posn	Pass	Chalk:1	2	3	4	5
L	6	1	TM Support	TM Support	TM Support	TM Support	T
L	7	1	TM Support	TM Support	TM Support	TM Support	T
L	8	1	TM Support	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D
L	9	1	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D
L	10	1	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D
L	11	1	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D
L	12	1	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D/2-325 AIR	D
L	13	1	TM Assault	TM Assault	TM Assault	TM Assault	
L	14	1	TM Assault	TM Assault	TM Assault	TM Assault	T
L	15	1	TM Assault	TM Assault	TM Assault	TM Assault	T
L	16	1	TM Assault	TM Assault	TM Assault	TM Assault	T
L	17	1	TM Assault	TM Assault	TM Assault	TM Assault	T
L	18	1	TM Assault	TM Assault	TM Assault	TM Assault	T
L	19	1	TM Assault	TM Assault	TM Assault	TM Assault	T
L	20	1	TM Assault	TM Assault	TM Assault	TM Assault	T
L	21	1					

23	0	0	0	7	0	0	7	7
24	0	0	0	7	0	0	7	7
25	4	0	0	3	0	0	7	7
26	7	0	0	0	0	0	7	7

Figure 12. Analytical Optimal Solution Display. In this example the left door, position 6, first pass of aircraft number 1 is occupied by a jumper from TM Support.

Hybrid Model Interface

In order to include the inherent variability into the problem, the analytical model solution is provided as the input for the simulation model. This is done in the user application by establishing a series of text files, which are the initialization parameters for the simulation model. The components that size and determine the parameters of the analytical model are carried out in the simulation model as well. The data elements communicated from the analytical to the simulation model are as follows:

- Type and number of chalks (i.e. aircraft)
- Personnel Point of Impact X and Y location
- Aircraft heading in degrees
- Number of jumpers per pass
- Number of passes
- Total number of jumpers on each door
- The racetrack time (i.e. go around again time)
- X and Y locations of each objective
- The objective names, units and force assigned
- The ground speed over given terrain to each objective
- A left door manifest by unit
- A right door manifest by unit

For data accuracy, the different elements for the objectives are stored in separate text files. One file contains the names, another the units responsible, and other files store the locations, force requirements, and relative maximum speed over the specified terrain. Each line of data is read by the simulation model at initialization, and dynamically re-sizes the model accordingly. It is here that an object-oriented simulation shows its greatest advantage, being re-sized without recompiling, and existing in a runtime form. The global variables of the analytical model become the global variables of the simulation.

Simulation Model

CACI's MODSIM II for Windows, object-oriented simulation software, was selected as the implementation package for this application for several reasons:

1. Once compiled, the runtime executable file runs in Windows 3.1 and is royalty free.
2. It can readily import the optimal manifest and initialization parameters generated by the optimization package in Visual Basic.
3. It is object-oriented, and can track the several hundred entities and their actions with less memory requirements than other packages.

The simulation model uses an inherited *stream object* that both reads and writes from text files, and sets each entry equal to a variable value. The global values are the attitude and orientation of the aircraft, the X and Y locations of the objectives, as well as the force requirement and designation assigned to each of the objectives. The initialization parameters are identical to the Visual Basic application (i.e. the mathematical model). Both the left and right door manifests are read and stored in corresponding two-dimensional arrays for position by chalk number. Thus the entire simulation model is sized and initialized, without any additional user input. All required data is read from the analytical model application. A unique communication object is responsible for transmitting all global variable data to every software module within the simulation model.

The objects generated by the simulation model represent the real life objects in the operation. First, both a left and a right door are created as a type of *queue object*. These door objects "fly" side by side, but disperse jumpers independently. The aircraft door objects are stored in an array of objects for easy selection, designation, and tracking. The jumpers are created from number one through the aircraft capacity and are given a unit designation according to their position in the manifest array. The jumpers are then added to their corresponding chalk door queue. At jumper initialization, the random number object that generates the random numbers for that object is also initialized and accompanies it throughout the simulation. The door objects dispense the jumpers in a first come, first served (FIFO) order. Finally, the objectives and their characteristics are generated as *statistical objects* for monitoring, and are also stored as an array of objects. Figure 13 shows the object model diagram for the primary objects used in the simulation.

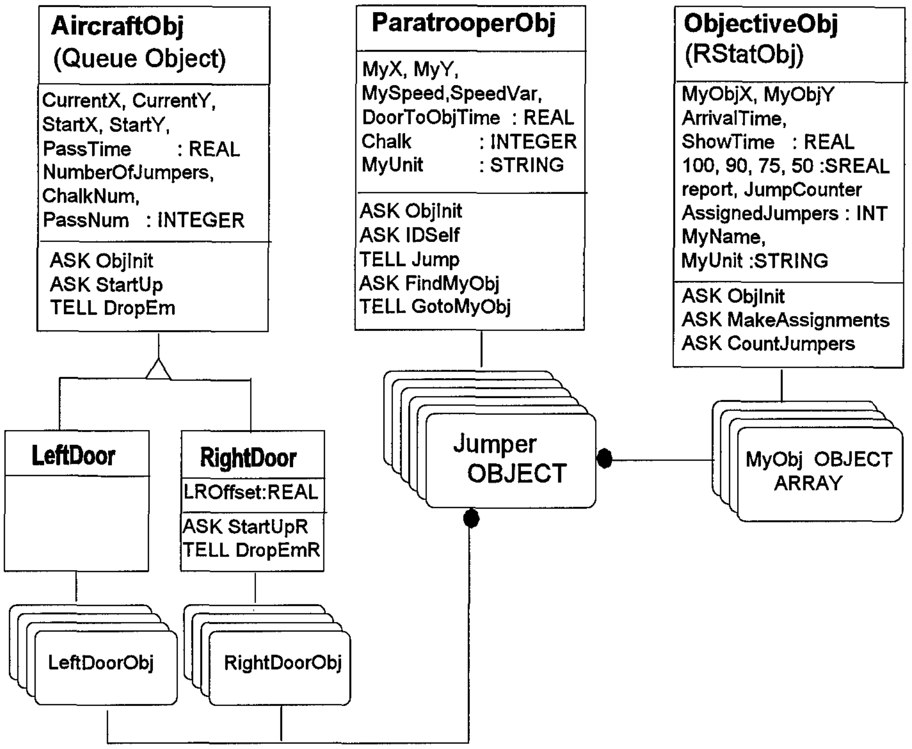


Figure 13. Simulation Object Model.
Format [Rumbaugh, 1991]

The main purpose for running the simulation model, is to introduce the inherent variability present in the problem. This includes the accuracy of the aircraft crew in selecting the Calculated Air Release Point (CARP), hitting the Personnel Point of Impact (PPI) location accurately, flying at the specified speed and direction, and attaining the required altitude. Subsequently, the jumper start times, jump interval between left and right door, same-door jumper interval spacing, and rates of fall are subject to variance. Upon impact with the ground, each jumper independently de-rigs (gets out of his harness and places his weapon into operation), prepares equipment, orients towards his objective, and moves to that objective, all with observable distributions. Each instance includes an amount of variance that is used in the simulation model, and is shown in Table 3. The distributions were selected by entering the raw data into BestFit v1.12, and ranking the MODSIM available distributions using Kilmogorov-Smirnov test parameters. To illustrate, jumper interval had the best fit among all the available distributions using a gamma distribution, which makes sense because it can be viewed as a queue's inter-arrival time. See Appendix C for additional statistical data on how the distributions are selected. Note that whenever historical data did not exist, or was inaccessible, estimates from experienced field experts were used.

Table 3. Simulation Statistical Distributions

Behavior	Distribution Selected	Shape Parameters

Aircraft Speed (Knots)	Gamma	alpha = 496.0
		beta = 0.25

Direction of Flight	Triangular	-10, 0 , +10 degrees
StartX, StartY	Triangular	-240, 0 , +240 meters

Left/Right Door Offset	Normal	mean = 1.54381 sec.
		sigma = 0.259065

Jump Interval (seconds)	Gamma	alpha = 11.04
		beta = 9.43e-2

Deployment Altitude	Normal	mean = 679.3 ft.
		sigma = 23.49

DriftX, DriftY	Triangular	+/- DeployAlt * 3.841/100 meters

Fall Time	Normal	mean = 39.2 second
		sigma = 4.9

Ground Speed Variance	Normal	mean = 1.0
		sigma = 0.25

De-rig Time	Normal	mean = 7.0 minutes
		sigma = 1.5

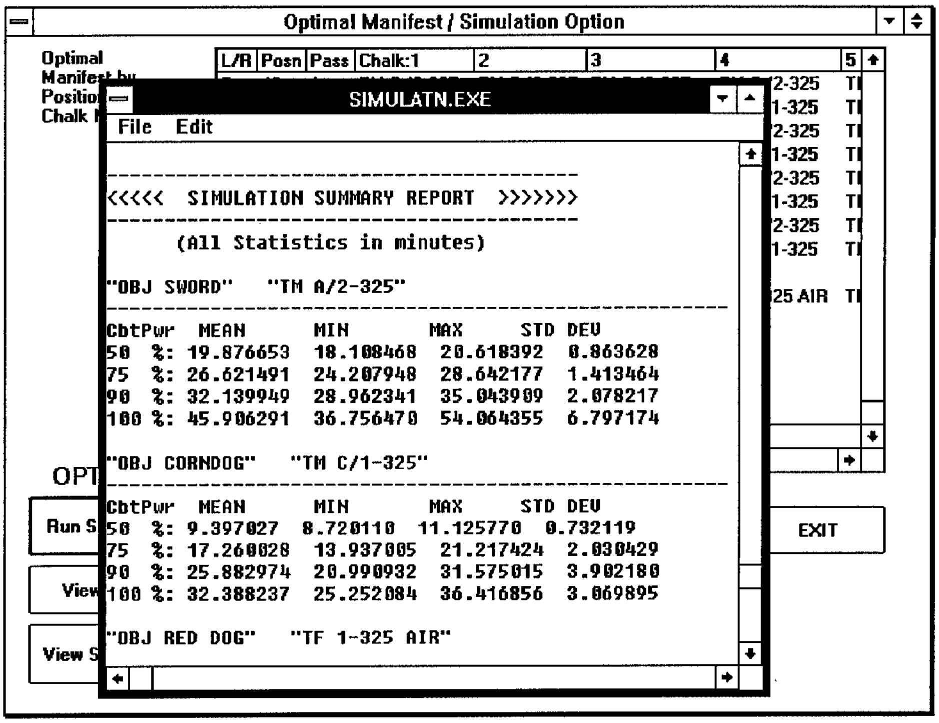
Once the simulation model is initialized and all objects are created, the simulation can commence. First, each aircraft is assigned a start point around the CARP. Then for each pass, the aircraft, in chalk order, begin to drop the jumpers. The drop is accomplished by removing the first jumper in the queue, and "telling" it to jump. The aircraft door object then "flies" for a jump interval duration, updates its location, and then discharges the next jumper, and so on. The non-primary right jump door, begins its operations after the offset interval has been flown. Subsequent chalks begin 8 to 10 seconds after the previous chalk. The difference between primary and alternate door initiation averaged 1.611 seconds, as opposed to the planned 1/2 second, with a standard deviation of 0.329 seconds. The delta between jumpers exiting the aircraft has a mean of one second (1.02) with a standard deviation of 0.327 seconds. Each aircraft discharges the number of jumpers designated for each pass in sequence until the aircraft objects are empty and disposed of.

At jump time, each jumper object receives the current *x* and *y* location and altitude from the aircraft, calculates the amount of induced throw and drift, and waits the required fall time. After the fall time has been delayed, the jumper automatically looks for his objective by matching his "*MyUnit*" name, with the objective's "*MyObjUnit*" name from the objective name array, and stores the matched objective number. The objective number is used in the ensuing method, when the jumper object computes the distance and time required to reach his objective. The distance for each jumper is calculated by selecting the objective *x* and *y* locations from the arrays and using Pythagorean's Theorem. The distance is then divided by the speed from the objective speed array, multiplied by a random path factor (to compensate for straight line distance and broken terrain) and converted into time in seconds by dividing by the estimated speed over the given terrain. The jumper object then waits the determined de-rig time, followed by his movement time, and then reports to his objective.

The objectives are persistent statistical objects and maintain the eventual output data throughout each replication of the simulation. Once the jumper reports to the objective, the objective records its arrival time as established by the simulation clock. The jumper object is then disposed of. The instance of each arrival is counted and then the counter value is divided by the required number of jumpers, yielding a current percentage value of the reported jumpers over those assigned. This percentage is the combat power for that object at that moment in time. When this combat power reaches 50, 75, 90, and 100 percent, the times are recorded for future output and analysis. The build up of combat power is an essential statistic for commanders in order that they make timely decisions. For example, an enemy strong point may only be able to be secured if a certain minimum combat power is present. The actual percentages used by airborne units are translated into codewords in order that they may be broadcast over an unsecure means if necessary. These codewords are used as a quick means of updating the commander on his subordinate units present condition.

Also note that all jumpers with a null entry (space key) for unit are disposed of immediately upon exiting the aircraft, as their statistics are not relevant to the purpose of the simulation. However these null jumpers must still occupy the manifest “space” in order to accurately displace those jumpers that are tracked.

After the replications of the simulation are complete, the combat power build-up statistics are recorded and displayed to the user. These statistics include the expected time (mean), and the best and worst instances of reaching those levels at each objective. All replication results and the simulation statistics are viewed within the simulation window inside the original Windows application. See Figure 14 for a sample of the simulation results in the window. The complete MODSIM simulation code is included in Appendix B. In the next chapter, the verification and validation of the hybrid model, as well as a preliminary comparison to present methods is discussed.



```
----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
----------------------------------------
      (All Statistics in minutes)

"OBJ SWORD"   "TM A/2-325"
---------------------------------------------------------
CbtPwr  MEAN       MIN        MAX      STD DEV
50  %: 19.876653  18.108468  20.618392  0.863628
75  %: 26.621491  24.207948  28.642177  1.413464
90  %: 32.139949  28.962341  35.043909  2.078217
100 %: 45.906291  36.756470  54.064355  6.797174

"OBJ CORNDOG"   "TM C/1-325"
---------------------------------------------------------
CbtPwr  MEAN       MIN        MAX      STD DEV
50  %: 9.397027  8.720110  11.125770  0.732119
75  %: 17.260028  13.937005  21.217424  2.030429
90  %: 25.882974  20.990932  31.575015  3.902180
100 %: 32.388237  25.252084  36.416856  3.069895

"OBJ RED DOG"   "TF 1-325 AIR"
```

Figure 14. Simulation Model Outputs

Example Scenario

An example scenario is developed in order to present the initialization of data elements through the solution procedure. The output of the analytical model is the optimal manifest listing, given the conditions, and the output of the simulation model is the expected results. The scenario to be used in this example is a relatively common battalion-sized training mass tactical operation taking place on Sicily Drop Zone, Fort Bragg, North Carolina. Five C-141 aircraft are used, dropping 120 paratroopers each on a single pass. Paratroopers are assigned one of six objectives. The initialization parameters that must be entered by the user can be seen in Table 4. The environmental factors are uniform for each objective in this scenario.

Once the parameters have been entered, the transportation network is formed and solved. After the last objective data input screen has been completed, the optimal

Table 4. Example Scenario Initialization Parameters

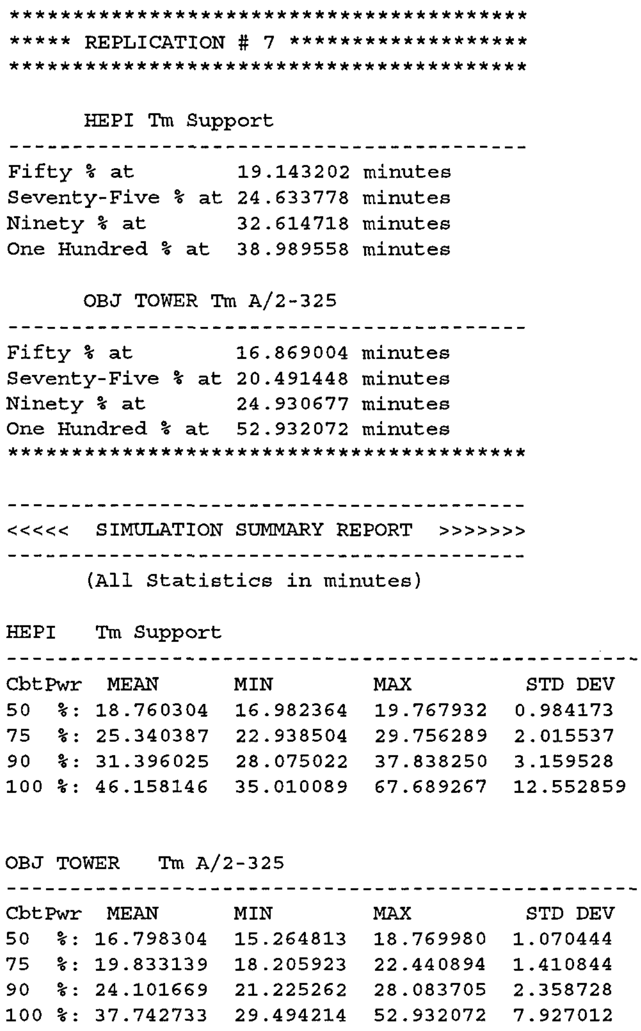
Data Element	Value
Number of Aircraft	5
Type of Aircraft	C-141
Length of DZ	62 seconds
Direction of Flight	210
PPI Location	70500 92000
PPI Altitude	237 meters
Number of Passes Planned	1
Jumpers per Door per Pass	60
Number of Objectives	6
Objective 1 Data:	
Name	HEPI
Unit	Tm Support
Location	70400 91900
Assigned	76
Objective 2 Data:	
Name	OBJ TOWER
Unit	Tm A/2-325
Location	69750 91800
Assigned	108
Objective 3 Data:	
Name	OBJ SNOW
Unit	Tm B/2-325
Location	69300 91100
Assigned	96
Objective 4 Data:	
Name	OBJ FALCON
Unit	Tm C/2-325
Location	67350 91250
Assigned	116
Objective 5 Data:	
Name	OBJ GREEN
Unit	Tm Breach
Location	69900 90100
Assigned	104
Objective 6 Data:	
Name	OBJ BLOCK
Unit	D/2-325
Location	68900 88000
Assigned	76

manifest by position and chalk number is presented. The manifest summary is shown in Table 5, where the chalks are on the left, with the units across the top, and the positions are designated for both the left and right door. From this screen, the user may either review the time coefficients on another screen, view the final tableau solution, or chose to simulate the generated manifest. To start the simulation, the user simply has to press the "Run Simulation" button from the options. Once the simulation has been selected, a shell window is opened and the simulation is executed within this window.

Table 5. Manifest Summary Table

Chalk Number	Unit: Tm Support	Tm A/2-325	Tm B/2-325	Tm C/2-325	Tm Breach	D/2-325
1	1..8	9..18	19..28 39..40	41..50	29..38	53..60
2	1..8	9..18	19..28	39..50	29..38	53..60
3	1..8	9..18	19..28	39..50	29..38	53..60
4	1..8	9..18	19..28	39..50	29..38	53..60
5	1..6	7..20	21..26	39..50	27..38	55..60

The simulation first reads all its initialization parameters from the text files, and generates all the aircraft, objectives and jumpers with the proper attributes. The simulation then reports the build-up of combat power times at each objective for each replication. A summary table is printed after all the replications have been run that displays the maximum and minimum observed values, the mean and the standard deviation for each objective and percentage value. See Figure 15 for a replication report and the final report format. Upon completion of the simulation, the user then records the data and exits the model.



```
*****************************************
***** REPLICATION # 7 *******************
*****************************************

      HEPI Tm Support
-----------------------------------------
Fifty % at         19.143202 minutes
Seventy-Five % at  24.633778 minutes
Ninety % at        32.614718 minutes
One Hundred % at   38.989558 minutes

      OBJ TOWER Tm A/2-325
-----------------------------------------
Fifty % at         16.869004 minutes
Seventy-Five % at  20.491448 minutes
Ninety % at        24.930677 minutes
One Hundred % at   52.932072 minutes
*****************************************

-----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
-----------------------------------------
      (All Statistics in minutes)

HEPI   Tm Support
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX         STD DEV
50  %: 18.760304  16.982364  19.767932  0.984173
75  %: 25.340387  22.938504  29.756289  2.015537
90  %: 31.396025  28.075022  37.838250  3.159528
100 %: 46.158146  35.010089  67.689267  12.552859

OBJ TOWER   Tm A/2-325
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX         STD DEV
50  %: 16.798304  15.264813  18.769980  1.070444
75  %: 19.833139  18.205923  22.440894  1.410844
90  %: 24.101669  21.225262  28.083705  2.358728
100 %: 37.742733  29.494214  52.932072  7.927012
```

Figure 15. Replication Results and Simulation Summary Table

Overall Solution Procedure

In this hybrid model, the analytical model develops a best-case solution for this class of problem. The simulation model not only serves to validate the analytical model, but more importantly, answers the second original question, “What can the commander expect to happen on the ground?”, when the variability is included in the model. Chapter 5 analyzes the accuracy of the hybrid model output, as well as utilizes the simulation model to compare the analytical solution versus the existing manual method.

CHAPTER 5
HYBRID MODEL VERIFICATION,
VALIDATION, AND
ANALYSIS OF RESULTS

In order for the developed hybrid analytical/simulation model to gain acceptance by the user, the analytical model must be verified to ensure that it is returning an optimal, or close to optimum solution compared to other solution methods. Secondly, the simulation model is used to validate the original assumptions made in the formulation of the analytical model (i.e. the *perfect world* conditions as described in Chapter 3). Lastly, the analytical model is validated by comparing the simulation results of the computer application package versus a manual, likely scenario.

Analytical Model Verification

In order to ensure that the Visual Basic application and the transportation network is returning a valid solution, the results of a sixty variable problem was compared to two other solvers. First, the problem was modeled and solved as a transportation network, with twenty source nodes (jumper location, with supply equal to the number of aircraft, i.e. five), three demand nodes (three objectives with 31, 29, and 26 jumper requirements) and a total of 100 jumpers. The arc costs were taken from a representative coefficient array of distances between the projected jumper impact locations and each objective. This problem was modeled and solved using Net Solve, a commercial network solver.

Microsoft's Excel 5.0's *Problem Solver* in the linear/integer programming mode was selected as a third method to solve this problem for comparison. Shown in Table 6, the total costs obtained from the Excel and the Net Solve solutions were the same, but their associated solution variable matrix varied slightly. The analytical model solution also reached an optimal solution with an identical total cost as the other two methods. The solution obtained from the analytical model agreed exactly with the Excel solution. The slight difference between the Net Solve and the analytical model solution matrices can be rationalized by examining the methodologies of eliminating the rows and columns from contention in the formulation of the initial basic feasible solutions. Net Solve uses the North West Corner Rule, while the analytical model uses Vogel's Approximation Method.

Table 6. Analytical Solution Method Verification and Comparison

Sample Coefficient Array

	A	B	C	Null
1	806	658	721	0
2	695	804	649	0
3	600	952	605	0
4	532	1101	597	0
5	500	1249	626	0
6	513	1398	688	0
7	566	1547	774	0
8	650	1697	877	0
9	755	1846	992	0
10	874	1996	1116	0
11	1000	2145	1245	0
12	1133	2295	1379	0
13	1269	2445	1515	0
14	1408	2595	1654	0
15	1550	2744	1795	0
16	1692	2894	1937	0
17	1836	3044	2081	0
18	1981	3194	2225	0
19	2126	3344	2370	0
20	2272	3493	2515	0

Optimal Seating Assignment from Excel Integer Program Solution

	A	B	C	Null
1		5		
2		5		
3		5		
4		5		
5		5		
6		4	1	
7			5	
8			5	
9			5	
10			5	
11	1		4	
12	5			
13	5			
14	5			
15	5			
16	5			
17	5			
18	0		1	4
19	0			5
20	0			5
	Cost:	101540		

Optimal Seating Assignment from NetSolve Transportation Solution

	A	B	C	Null
1		5		
2		5		
3		5		
4		5		
5		5		
6		4	1	
7			5	
8			5	
9			5	
10			5	
11	5			
12	5			
13	5			
14	5			
15	1		4	
16	5			
17	6			
18			1	4
19				5
20				5
	Cost:	101540		

Optimal Seating Assignment from Analytical Model Solution

	A	B	C	Null
1		5		
2		5		
3		5		
4		5		
5		5		
6		4	1	
7			5	
8			5	
9			5	
10			5	
11	1		4	
12	5			
13	5			
14	5			
15	5			
16	5			
17	5			
18	0		1	4
19	0			5
20	0			5
	Cost:	101540		

In terms of overall computational complexity, due to Net Solve's methodology for achieving an initial basic feasible solution, Vogel's method used by the analytical model application will be less complex for most problems. For the presented sample problem, the analytical model took 22 iterations to find an initial solution, and required 6 additional iterations of the transportation problem algorithm to reach the final solution. This process took 6.5 seconds of CPU time on a single 486DX2 processor, at 66 MHz. Excel problem solver required 92 transformations and 13.53 seconds of CPU time to solve the problem. Net Solve's number of iterations could not be recorded, and CPU time was 3.0 seconds (however, Net Solve's setup time was significantly longer than either of the other solvers, and its outputs could not readily be integrated into a simulation model). The savings in computational complexity and time by the analytical model is because the analytical model fully exploits the structure of the transportation network problem by using Vogel's Approximation Method and because it ignores the null jumpers (or the corresponding dummy objective node) until after the solution has been reached. Reference Appendix D for outputs of the Net Solve and Excel solutions.

Number Of Simulation Replications Required

In order to gain valid feedback from the simulation model, and be able to statistically compare strategies with a degree of accuracy, the proper number of simulation runs (or replications) required had to be determined. Due to the nature of this type of problem, with virtually an unbounded number of possible combinations of variables, the most likely factors were chosen to become effects in an experimental design that helped determine the required number of replications that yields statistically valid results. The experimental design model also conveys the relative importance of each of these effects, as well as their interaction (if any).

The important factors range from a simple case of a few objectives located relatively close together, to the extreme cases of many more objectives, and diverse ranges to each objective. The design and the conduct of the experiment are explained in the next section. The goal of this process is to examine the standard error of the mean at each objective, and determine if it is within 10% of the mean for those conditions. The equation for standard error is the standard deviation of the sample divided by the square root of the number of replications run. Each of the four experimental runs was iterated through until the standard error was within the required 10% of the mean.

At the point that the 10% condition is met, the research can then draw conclusions as to the required number of replications that must be represented in the simulation in order to be statistically valid for use in comparisons.

A two-factor Fisher experimental design was chosen to observe the standard error of the mean. The two factors and their high and low levels are:

1. Number of Objectives (3-Low, 8-High)

2. Relative Distance of Objectives from the aircraft line of flight (within 500 meters-Low, 1000 meters or greater-High)

The experiment runs and results can be seen in more detail in Appendix E. The experiment was conducted as a full factorial, with four runs. The measure of effectiveness for the sample scenarios is the build-up of 90% combat power at each objective, with each objective weighted equally. Ninety percent was selected as the most critical build-up percentage used by the commander for decision making, and avoids the outliers that are prevalent in the one hundred percent counter.

What was found in the experiment is that the standard error of the mean at all objectives was within 10% of the individual means for the fifty and seventy-five, and ninety percent power levels after seven replications. The data collected can be seen in Table 7. With as few as seven replications, the conclusion is made that this is the minimum required to represent the overall variability inherent in the system.

The experimental model derived from the simulation runs with the seven replications was analyzed and the main effects (the number of objectives and the relative distance) were very significant. There appeared to be virtually no two-way interaction. Because this was a full factorial design, the interaction effects were not confounded with either of the main effects. See the SPSS ANOVA table in Appendix E for the experimental effects of the model.

Table 7. Replication Experiment Results

		EXP #1					EXP #2	
Rep #	OBJ ONE	OBJ TWO	OBJ THREE		Rep #	OBJ ONE	OBJ TWO	OBJ THREE
1	25.302	27.390	22.402		1	40.398	54.043	29.799
2	22.897	30.728	23.61		2	36.81	60.759	40.295
3	24.534	22.685	25.81		3	41.24	44.726	34.13
4	22.067	23.080	25.390		4	38.372	45.885	34.767
5	23.17	24.337	25.422		5	40.006	49.085	35.666
6	21.78	23.490	29.026		6	35.858	44.206	37.602
7	21.90	27.17	28.10		7	36.267	55.268	36.91
means:	23.0956	25.5553	25.681		means:	38.4240	50.5674	35.5963
std dev	1.364	2.9684	2.318		std dev	2.167	6.281	3.273
Std Error	0.515	1.121	0.8762		***Std Error***	0.819	2.3742	1.237
10 %	2.3096	2.5555	2.568		10 %	3.8424	5.0567	3.5596
				EXP #3				
Rep #	OBJ ONE	OBJ TWO	OBJ THREE	OBJ FOUR	OBJ FIVE	OBJ SIX	OBJ SEVEN	OBJ EIGHT
1	17.35	12.88	21.48	13.60	13.99	18.93	18.38	22.852
2	19.02	14.95	19.25	14.18	14.07	19.35	16.14	25.16
3	19.08	14.01	19.19	12.50	15.94	15.39	19.79	23.687
4	20.479	14.18	17.39	13.75	16.62	15.26	17.22	22.874
5	18.13	13.31	15.47	16.22	15.64	16.23	18.68	25.466
6	17.85	12.11	17.07	14.78	16.21	16.96	19.68	22.935
7	19.30	13.80	20.983	13.16	15.00	16.86	20.423	27.16
means:	18.749	13.61	18.695	14.031	15.359	17.002	18.620	24.3066
std dev	1.049	0.931	2.170	1.206	1.032	1.606	1.520	1.667
Std Error	0.3965	0.352	0.8203	0.456	0.3904	0.6073	0.5748	0.6302
10 %	1.874	1.361	1.869	1.403	1.535	1.700	1.862	2.4307
				EXP #4				
Rep #	OBJ ONE	OBJ TWO	OBJ THREE	OBJ FOUR	OBJ FIVE	OBJ SIX	OBJ SEVEN	OBJ EIGHT
1	32.866	39.81	25.837	25.308	40.634	27.657	40.426	43.264
2	34.325	41.67	32.753	25.356	41.89	27.936	44.273	43.785
3	34.258	36.276	29.13	21.37	33.565	33.00	42.362	43.81
4	35.452	36.232	27.364	26.11	37.743	40.054	36.930	39.769
5	35.245	33.71	29.707	27.889	34.888	30.596	37.324	36.033
6	28.260	40.478	27.263	24.852	37.81	34.497	39.329	45.294
7	31.91	42.18	27.553	26.080	36.13	29.450	45.897	56.377
means:	33.188	38.6253	28.515	25.281	37.5259	31.884	40.9344	44.0479
std dev	2.510	3.218	2.2639	1.981	2.9854	4.3944	3.4088	6.2830
Std Error	0.9488	1.216	0.8557	0.749	1.128	1.660	1.288	2.3747
10 %	3.318	3.8625	2.851	2.5282	3.7526	3.188	4.0934	4.4048

Validation Of Analytical Model Assumptions

In order to bound the original problem and derive an analytical solution procedure, several assumptions were made in Chapter 3. The simulation portion of the hybrid model can be used to test the validity of many of these original assumptions, mainly those dealing with the aircraft heading, proper PPI accuracy, and operation initialization timing. For this validation, a sample problem is presented, solved by the analytical model, and then simulated with 7 replications in the simulation model. The results of this solution method and the corresponding optimal manifest are compared to simulation results in which the initialization parameters have been altered to reflect degradation of aircraft related assumptions (also using the same manifest each time).

For this validation procedure, the same example scenario is used for each comparison. The ranges of allowable variance before the optimal solution would require significant change in the manifest order is directly attributable to each possible scenario. The following scenario is presented as a relatively simple example that can demonstrate the sensitivity of the solution method to large degrees of variance. Table 8 shows the scenario initialization parameters. Appendix F contains all simulation run summary tables as well as hypothesis testing for the three instance shown here. A pairwise t-test is used to directly compare the jumper arrival times at their appointed objectives of the new scenarios against the standardized case presented. Note that the simulation is re-initialized for each scenario run and a common random number string is used for each identical portion of the scenarios [Law, Kelton, 1991]. The equations for a paired t-test mean, variance and the confidence interval are as follows:

mean: $$\overline{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}$$

Z = the delta between the baseline and the test
n = number of the sample
j = the counter

variance: $$\hat{Var}[\bar{Z}(n)] = \frac{\sum_{j=1}^{n}[Z_j - \bar{Z}(n)]^2}{n(n-1)}$$

confidence interval: $$\bar{Z}(n) \pm t_{n-1,1-\alpha/2}\sqrt{\hat{Var}[\bar{Z}(n)]}$$

The tests conducted are to see if zero falls within the confidence interval, if so, the hypothesis that the two scenarios are the same cannot be rejected. Therefore, if zero is in the interval for the deviation, the amount of deviation induced is insignificant to the model. The following are the three analytical model validation experiments.

Table 8. Comparison Scenario Initialization Parameters

Number of Aircraft	1
Aircraft Type	C-130
PPI Location (grid)	70500 92000
Direction of Flight	202 Degrees
Number of Objectives	3
Objective #1 (Mission Essential) Unit Name Location Requirements	 Tm Assault OBJ DAGGER 69800 91900 20
Objective #2 (Mission Support) Unit Name Location Requirements	 Tm Secure OBJ KNIFE 69500 91000 20
Objective #3 (Other Support Mission) Unit Name Location Requirements	 Tm Support OBJ SWORD 70000 90000 20
Environmental Conditions (includes all objectives)	
Temperature	Mild
Humidity	50%
Cloud Cover	Night
Grade	Level
Surface	Hard Dirt

Late Operation Initiation

One of the most likely events that can happen in an airborne operation is that the aircraft crew can over-fly the CARP by several seconds, delaying the lighting of the green light that allows jumpers to exit. By comparing the build up of 75% combat power at each objective from the simulation runs, the analysis is made that there is no significant alteration in mean assembly times until more than 2.0 seconds has elapsed. This 2.0 seconds also translates to 140 meters on the ground. Therefore the original assumption is valid, with this scenario set-up, as long as the green light is lit within 2.0 seconds of its designated calculated air release point. See Table 9 for the confidence intervals.

Lateral Deviation in CARP

Lateral deviations can occur when the aircraft flies the intended heading, but does not fully line-up on the designated CARP. This lateral drift can also occur when there is a strong unforcasted cross-wind, perpendicular to the direction of flight. By altering the PPI location in subsequent simulation runs by up to 150 meters, the original manifest remains valid with a 90% level of confidence. This 150 meter lateral shift is also equivalent to an unforcasted 4.0 knot wind shift, which is one-third of the allowable ground wind speed to conduct parachute operations (e.g. jumps will be waived-off if winds exceed 12 knots). The confidence interval from the pairwise t-test is found in Table 9.

Direction of Flight Variation

According to current practice of U.S. Air Force Air Mobility Command pilots, a pass will be aborted if they fly more than 5 degrees left or right of the designated direction of flight. In the comparison of the control optimal solution results and running different alterations of heading, it was found that heading could indeed vary up to 5 degrees, and still not significantly affect the solution results with a 90% level of confidence. The confidence interval established for each of the three cases can be seen in Table 9.

Table 9. Paired t-Test 90% Confidence Intervals

Trial	Upper Bound	Lower Bound
Late Green Light	0.365751	-0.00461
Shifting the PPI by 150 meters	0.066851	-0.58368
5 Degrees Off Direction of Flight	0.149303	-0.23161

Now that it has been demonstrated that the verifiable analytical model original assumptions were indeed valid, a direct comparison is made using a manual solution for a real scenario and the computed analytical/optimal solution manifest.

Analytical vs. Manual Solution

A full test scenario is developed and solved by the analytical model returning a solution manifest, which is then compared against a manually prepared manifest solution.

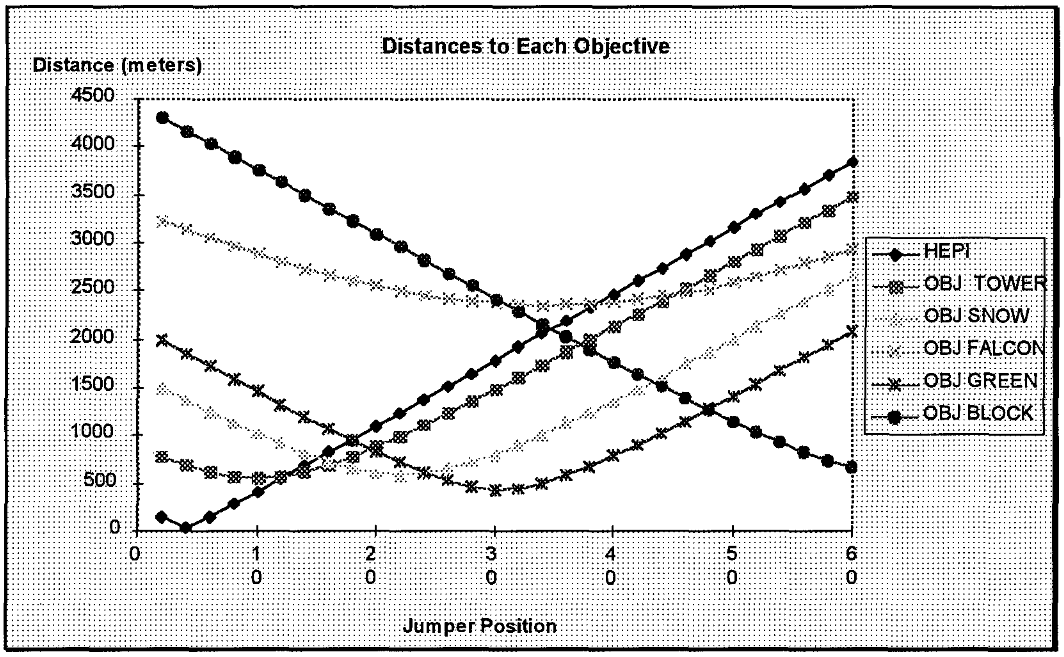


Figure 16. Graph of Distances to Each Objective from Impact Locations

The scenario for this comparison is established with five C-141 aircraft and six objectives. The objectives are assigned importance factors by the commander and can be found in the summary Table 10. The distance coefficients from the projected impact points to each of the objectives is graphed in Figure 16. As seen in the graph, it is up to both the manual and analytical model to find the shortest travel distances and place as many jumpers as possible in those positions in the manifest.

The manual solution includes the insertion of headquarters personnel into lead exit positions, as well as a different distribution of jumpers in the aircraft, primarily focusing on the mission essential objective. A breakdown of each manifest can be seen in Appendix F. Both solution scenarios were run through the simulation model with seven replications each, again using a common random number string for each instance of variance. The jumper arrival times at each objective was then collected and directly compared using a pairwise t-test.

It can be said with 90% confidence, that the analytical model resulted in a superior solution than the manual method, as well as saving a great deal of time. This is reflected in the results shown in Table 10. Three of the six objectives were seized faster using the analytical method, two were quicker by the manual method, with one objective's test inconclusive (i.e. zero in the confidence interval). Using the qualitative points established by the commander's priorities in the scenario, the analytical model scores 6.9 total points versus 6.2 for the manual method. The reason the manual solution prevails for two of the objectives is that emphasis had been placed upon minimizing the travel distance for the mission essential task, rather than for the entire operation as done by the analytical model.

Table 10. Analytical vs. Manual Paired t-test Analysis

OBJECTIVE	Paired Delta Mean	Variance	90% Confidence Interval		Conclusions:	Points
OBJ HEPI	10.161	0.404	11.207	9.115	Analytic Superior	2.3
OBJ TOWER	-2.193	0.091	-1.696	-2.690	Manual Superior	5.2
OBJ SNOW	-0.127	0.099	0.391	-0.644	Not Conclusive	1.0
OBJ FALCON	-4.842	2.571	-2.205	-7.480	Manual Superior	1.0
OBJ GREEN	12.797	0.179	13.493	12.101	Analytic Superior	2.3
OBJ BLOCK	0.801	0.183	1.505	0.097	Analytic Superior	2.3

While this is in fact just one example operation, the conclusion is drawn that for other possible combinations, this hybrid model solution should remain valid. There may exist extreme values for mission priority, large mix of objective relative distance, as well as jumper unit mix, and some may not yield a solution that is what the commander would want. It is in these rare cases that the commander would analyze the simulation output and find the combat power build-up statistics are unsatisfactory for one or more mission critical objective. It is then left to the user to prepare a manual manifest, and iterate through the simulation model, until a suitable solution is found, or discover that the parameters must be changed to accomplish the mission. In this same scenario, a sensitivity analysis of weights was conducted. The analysis concluded that if the mission weights were changed to 12.0 for mission essential, and 4.0 for mission support, that the analytical solution was equal to or surpassed the manual solution for every objective.

For each instance of comparison and experimentation, the analytical portion of the hybrid model was shown to yield a relatively robust optimal solution (i.e. could accept a degree of variability). The simulation portion of the model also proved quite advantageous in comparing multiple strategies and conditions. Overall the hybrid analytical/simulation model appears valid for this class of problem.

CHAPTER 6
CONCLUSIONS AND SUMMARY

This chapter summarizes the research presented in previous chapters and discusses its significance to the area of hybrid modeling.

Mass tactical airborne operations represent a large complex problem that had not yet been targeted for optimization because it was thought to have too many parameters and too many instances of independent variance. The aim of the commander is to load his aircraft in such a way that the minimum amount of time is required to seize all assigned objectives. The personnel loading portion of the air plan, or manifest, is directly attributable to where each jumper should impact the ground. The manifest is a by unit listing of *who* occupies *what* seat in *which* aircraft. The jumpers will exit in exactly the order loaded. It is the aim of this research to apply a hybrid analytical/simulation model to this problem in order to minimize the total travel distance and time required of each paratrooper. Secondly, the simulation model provides the commander feedback as to what to expect upon mission execution.

The procedure followed by this model first establishes the mission parameters in a mathematical model. This math model disregards the possible variations and presents a *perfect world model* where the aircraft flies at the exact specified heading, speed and altitude, the Calculated Air Release Point (CARP) is properly computed to compensate for wind conditions, the jumpers exit on time, and at a uniform interval. Upon exiting the aircraft, it is assumed the jumper would be thrown to his exact impact location. Distances between impact points and the specified objectives are straight-line only (i.e. the world is flat with no obstacles, and no one gets lost). The mathematical model projects the assumed impact points of each jumper along a vector parallel to the line of flight originating at the designated personnel point of impact (PPI). The distances between the impact points and each designated objective are calculated using Pythagorean's theorem, which is then divided by a speed estimate, in meters per second, that translates the distances to time in seconds. Each time value associated with a particular objective is then multiplied by a commander's importance constant.

The preceding perfect world model is modeled as an assignment network, and solved very efficiently using a transportation problem algorithm. The impact locations become the supply nodes, and the objectives become demand nodes with an assignment requirement. The supply and demand nodes are connected using capacitated directional arcs. The solution method for transportation problems deals strictly with discrete integer values (i.e. jumpers and aircraft), without having to deal with the complexities of integer programming. Vogel's Approximation Method is used to calculate an efficient initial basic feasible solution, and the transportation algorithm iterates until an optimal condition is achieved. The outputs of the analytical solution procedure is the optimum manifest listing by chalk and unit of every paratrooper that will participate in that mission.

While this analytical model yields an excellent starting solution, the world is indeed not perfect and paratroopers and aircraft are subject to great amounts of induced variability. At this point a simulation model is required to introduce this variance back into the overall model. The simulation model is object-oriented and initialized using the identical parameters that established the analytical model. No additional user inputs are required to start the model. An object-oriented simulation model was selected as it is able to efficiently track the characteristics of a large number of objects simultaneously. The object-oriented model very closely follows the actual actions found in these type of operations.

In the simulation, aircraft stray off their predetermined heading, altitude and speed, and may even delay green light procedures. Jumpers are subject to wind drift and varying rates of descent. Upon impact, each jumper must de-rig his equipment individually, locate his objective, and move towards it at an individual rate of speed.

The objective objects in the simulation collect the build-up of combat power statistics over time. After each replication, the objective build-up times are presented, and upon completion of the replications, a summary table is output. The summary table provides an analysis of each objective, the best and worst observed arrival percentages, as well as the standard deviation. The simulation model not only provides a framework for comparison of alternatives and analytical model testing, but most importantly provides the commander a range of expected values that can be used in his overall command estimate process.

The entire hybrid model was coded in two separate software languages, Visual Basic 3.0 for Windows for the analytical/mathematical model, and MODSIM II object-oriented simulation software for the simulation model. All parameters are input in the Visual Basic shell, the manifest optimized, with all required entries automatically extracted for use by the simulation. The simulation then returns its output through the original window. The entire hybrid application is stand alone and royalty free.

From this research, it has been shown that a hybrid analytical/simulation model can be used to represent and solve a complex problem. The solution presented in the example scenario was indeed superior to one generated by the manual method, with a substantial overall savings in time and effort as well. The closed form analytical model can be solved quickly, and used in conjunction with the simulation model to present an overall solution procedure.

There also seems to be an added benefits of the application of analysis to the problem, other than solving for the optimal manifest. The greatest indirect benefit is that the process forces the airborne commander to examine, in detail, the priorities for his assault units and objectives. These qualitative questions are now directly answered by the individual responsible for the actions on the ground, and directly translates toward the air manifest in the process. This research also demonstrated the value of including simulation models into a form of decision support system.

The overall solution is very robust, and provides a means for feedback and analysis. Hybrid modeling techniques are a reminder that an analyst should not just limit themselves to a single model representation, but possibly seek an optimal set of models for a system, given the conditions.

REFERENCES

Adomian, George, “Solution of Complex Dynamic Systems”, Simulation, Vol. 54, No. 5, May 1990, pp. 245-251.

Armacost, Andrew, S. Mehrotra, “A Computational Comparison of the Network Simplex Method with the Dual Affine Scaling Method”, OPSRESEARCH, Vol. 28, No. 1, 1991, pp. 18-35.

Cavanaugh, K J, "Multilevel Approach to Minimum Cost Network Flows", Naval Postgraduate School Master's Thesis, September 1992.

Evans, James R., E. Minieka, Optimization Algorithms for Networks and Graphs, Marcel Dekker Inc., 1992.

Fang, Shu-Cherng, S. Puthenpura, Linear Optimization and Extensions: Theory and Applications, Prentice Hall, New Jersey, 1993.

Fredrickson, Greg N. “A Note on the Complexity of a Simple Transportation Problem”, SIAM Computing, Vol. 22, No. 1. February 1993, pp. 57-61.

Gilmer, John B., J. Adams, “Managing Uncertainty in Simulation”, 62d Military Operations Research Symposium, June 1994.

Glover, Fred, D. Karney, D. Klingsman, N. Napier, “A Computation Study on Start Procedures, Basic Change Criteria, and Solution Algorithms for Transportation Problems”, Management Science, Vol. 20, No. 5, January 1974, pp. 793-813.

Hanssman F., G. Dinif, W. Fischer, and S. Ramer, “Analytical Search Model for Optimum Seeking Simulations”, OR Spektrum, Vol. 2, 1980, pp. 91-97.

Hayes, Theodore R. “Dismounted Infantry Movement Rate Study” Simulation Technologies Letter Report 94-001, April, 1994.

Ignall, Edward J, P Kolesar, W E Walker, “Using Simulation to Develop and Validate Analytical Models: Some Case Studies”, Operations Research, Vol. 26, No. 2, March-April 1978, pp. 237-253.

Ignall, Edward J, P Kolesar, "On Using Simulation to Extend OR/MS Theory: The Symbiois of Simulation and Analysis", Current Issues in Computer Simulation, 1979, pp. 223-233.

Law, Averill M., W. D. Kelton, Simulation Modeling and Analysis, McGraw-Hill Inc., New York, 1991.

Keeler, Gerald Joseph, "A Hybrid Approach for Source Apportionment of Atmospheric Pollutants in the Northeastern United States", University of Michigan Ph.D. Dissertation, 1987.

Kimbleton, S., "A Heuristic Approach to Computer Systems Performance Improvement", AFIPS, Vol. 44, 1975.

Kirkwood, Craig W. "An Algebraic Approach to Formulating and Solving Large Models for Sequential Decisions Under Uncertainty", Management Science, Vol. 39, No. 7, July 1993, pp. 900-913.

Min, Moonkee, "A Knowledge-Based Hybrid Modeling Approach to Planning Problems in Flexible Manufacturing Systems", University of Michigan Ph.D. Dissertation, 1990.

Pascoe, Geoffery A. "Elements of Object-Oriented Programming", Byte, August 1986, pp. 15-20.

Patel, Himanshu M. "Technique Solves Transportation Problem", Industrial Engineering, March 1988, pp. 16-23.

Paskaramoorthy, Ratnam, "Hybrid Modeling of Elastic Wave Interaction with Inhomogeneities on an Elastic Medium", University of Manitoba Ph.D. Dissertation, 1990.

Rachev, S. T., L. Rüschendorf, "Solution of Some Transportation Problems with Relaxed or Additional Constraints", SIAM Journal of Control and Optimization, Vol. 32, No. 3, May 1994, pp. 673-689.

Romero, R, A. Monticelli, "A Hierarchical Decomposition Approach for Transmission Network Expansion Planning", IEEE Transactions on Power Systems, Vol. 9, February 1994, pp. 373-80.

Ruiz-Mier, Sergio, and J Talavage. "A Hybrid Paradigm for Modeling of Complex Systems", Simulation, Vol. 48, No. 4, pp. 135-141.

Rumbaugh, James, M. Blaha, W. Premerlani, F. Eddy, W. Lorenson, Object-Oriented Modeling and Design, Prentice Hall, New Jersey, 1991.

Sargent, Robert G. "A Historical View of Hybrid Simulation/Analytical Models", Proceedings of the 1994 Winter Simulation Conference, 1994, pp. 383-386.

Schwetman, H D, "Hybrid Simulation Models of Computer Systems", Communications of the ACM, Vol. 21, No. 9, September 1978, pp. 718-723.

Shanthikumar, J G, R G Sargent, "A Hybrid Simulation/Analytical Model of a Computerized Manufacturing System", Operational Research '81, 1981, pp. 901-915.

Shanthikumar, J G, R G Sargent, "A Unifying View of Hybrid Simulation/Analytical Models and Modeling", Operations Research, Vol. 31, No. 6, November-December 1983, pp. 1030-1052.

Taha, Hamdy A., Operations Research: An Introduction, MacMillan Press, New York, 1992.

Tzong, Tsair-Jyh, "Hybrid Modeling of Soil-Structure Interaction in Layered Media", University of California, Berkeley Ph.D. Dissertation, 1984.

Vemuri, V. Modeling of Complex Systems, Academic Press, New York, 1978.

Wagner, Harvey M. Principles of Operations Research, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

Waikar, Avinash, M. Helms, G. Graves, "A Framework for an AI-Based Hybrid Simulation System", The Industrial Robot, Vol. 20, No. 3, March 1993, pp. 20-26.

Whitaker, David, OR on the Micro, John Wiley & Sons, Chichester, 1984.

Winston, Wayne L., Introduction to Mathematical Programming, PWS-Kent Publishing Company, Boston, 1991.

82d Airborne Division Airborne Standard Operating Procedure (ASOP), Vol. 1, Edition IV, March 1990.

APPENDICES

APPENDIX A

VISUAL BASIC OPTIMIZATION MODEL CODE

```
Sub Form_Load ()
' defines the values for the Speed Array Look up table
' source of data is STI LR94-001
' speed expressed in meters per second.
'  Note:  Would like to do this more efficiently, but have not
'         figured out how ....
ReDim Speed(3, 3, 4, 3, 3) 'temp,humidity,cloud,grade,terrain

Speed(1, 1, 1, 1, 1) = 2.2   'Table D-2
    Speed(1, 1, 1, 2, 1) = 2.4
    Speed(1, 1, 1, 3, 1) = 1.5
    Speed(1, 1, 1, 1, 2) = 2.1
    Speed(1, 1, 1, 2, 2) = 2.3
    Speed(1, 1, 1, 3, 2) = 1.4
    Speed(1, 1, 1, 1, 3) = 1.45
    Speed(1, 1, 1, 2, 3) = 1.65
    Speed(1, 1, 1, 3, 3) = .85
    Speed(2, 1, 1, 1, 1) = 1.95   'Table D-4
    Speed(2, 1, 1, 2, 1) = 2.15
    Speed(2, 1, 1, 3, 1) = 1.3
    Speed(2, 1, 1, 1, 2) = 1.85
    Speed(2, 1, 1, 2, 2) = 2.05
    Speed(2, 1, 1, 3, 2) = 1.2
```

Majority of Table Ommitted to Save Space

```
    Speed(3, 3, 4, 2, 1) = .65    'Table G-34
    Speed(3, 3, 4, 2, 1) = .85
    Speed(3, 3, 4, 3, 1) = .25
    Speed(3, 3, 4, 1, 2) = .6
    Speed(3, 3, 4, 2, 2) = .8
    Speed(3, 3, 4, 3, 2) = .25
    Speed(3, 3, 4, 1, 3) = .4
    Speed(3, 3, 4, 2, 3) = .6
    Speed(3, 3, 4, 3, 3) = .15

End Sub

Sub Help01_Click ()

    frame1.Visible = True

End Sub

Sub InputObj_Click ()
    GDoF = Heading.Text
    GPPIX = PPIX.Text
    GPPIY = PPIY.Text
    GPPIAlt = PPIAlt.Text

ReDim Objectives(1 To GNumObj + 1)' establishes size of objective struct
```

```
'Creation of X and Y Grnd Location Arrays
    ReDim GXLoc(1 To GJumpPass * 2)
    Dim XL As Integer
    Dim ThrowX As Single
    For XL = 1 To GJumpPass * 2
      GXLoc(XL) = GPPIX + (Sin(GDoF) * ThrowX)
      ThrowX = ThrowX + 34.29
    Next XL

    ReDim GYLoc(1 To GJumpPass * 2)
    Dim YL As Integer
    Dim ThrowY As Single
    For YL = 1 To GJumpPass * 2
      GYLoc(YL) = GPPIY + (Cos(GDoF) * ThrowY)
      ThrowY = ThrowY + 34.29
    Next YL

' Sets default GNumJumpDoor at ACType values
    If GACType = 1 Then GNumJumpDoor = 60
    If GACType = 2 Then GNumJumpDoor = 30
    If GACType = 3 Then GNumJumpDoor = 50

' sets GNumJumpPass equal to number in the pass if only one pass
    If GPasses = 1 Then GNumJumpDoor = GJumpPass

' disables done and help buttons on manifest form
 InputObj.Enabled = False
 Help01.Enabled = False

' write neccesary info to info.txt file
Open "c:\manifest\INFO.txt" For Output As #1
    Write #1, GDoF
    Write #1, GPPIX
    Write #1, GPPIY
    Write #1, GChalks
    Write #1, GACType
    Write #1, GPasses
    Write #1, GJumpPass
    Write #1, GRace         'return '0' if not used
    Write #1, GNumObj
Close #1

' Goes to the Verify Number of jumpers screen
    If GNumObj >= 1 Then
      Form4.Visible = True

    End If

End Sub

Sub JumpPass_Change ()
     GJumpPass = JumpPass.Text
     If GJumpPass > DZLength + 1 Then
```

```
      MsgBox "You have specified more jumpers per pass than can safely
exit, given the drop zone length", 48
    End If
End Sub

Sub NumObj_Change ()
     GNumObj = NumObj.Text
     InputObj.Enabled = True
End Sub

Sub Option1_Click ()
    GACType = 1
End Sub

Sub Option2_Click ()
    GACType = 2
End Sub

Sub Option3_Click ()
       GACType = 3
End Sub

Sub Passes_Change ()
    GPasses = Passes.Text
    If Passes.Text > 1 Then Label11.Visible = True
    If Passes.Text > 1 Then Text1.Visible = True
End Sub

Sub Text1_Change ()
    GRace = 10'default
    GRace = Text1.Text
End Sub

Begin Form ObjInfo

Sub Force_Change ()
    Objectives(1).MyObjForce = force.Text
End Sub

Sub Form_Load ()
    Combo1.AddItem "Cool-> Below 50°F"    'temperature ranges
    Combo1.AddItem "Mild-> 51°F to 75°"
    Combo1.AddItem "Warm-> Above 76°F"

    Combo2.AddItem "25 % and less"        'humidity
    Combo2.AddItem "50 %"
    Combo2.AddItem "75 % and higher"

    Combo3.AddItem "Night"                'cloud
    Combo3.AddItem "Clear Sky"
    Combo3.AddItem "Partly_Cloudy"
    Combo3.AddItem "Cloudy"
    Combo4.AddItem "Level"                 'grade
    Combo4.AddItem "Slopes Up"
    Combo4.AddItem "Slopes Down"

    Combo5.AddItem "Asphalt"               'terrain
    Combo5.AddItem "Hard Dirt"
    Combo5.AddItem "Loose Sand"

' size the distance array and the time array
ReDim GDistances(GJumpPass * 2, GNumObj + 1)
ReDim GTime(GNumJumpDoor * 2, GNumObj + 1)   'Net Coefficients

' define size of objective force array
ReDim GJumperSum(1 To GNumObj + 1)    'to include dummy

' define size and fill supply array GSupply
ReDim GSupply(1 To GNumJumpDoor * 2)
Dim fill As Integer
For fill = 1 To GNumJumpDoor * 2
    GSupply(fill) = GChalks
Next fill

' fills dummy portion of the array with zeroes
Dim X As Integer
For X = 1 To GJumpPass * 2
    GDistances(X, GNumObj + 1) = 0
Next X

End Sub

Sub NextObj_Click ()
' assign this objectives distance weight coefficient

Dim weight As Single
    If MsnEssnl.Value = True Then
        weight = GME: GMECount = GMECount + 1
    ElseIf MsnSpt.Value = True Then
        weight = GMS: GMSCount = GMSCount + 1
    ElseIf OSptMsn.Value = True Then
        weight = GOSM: GOSMCount = GOSMCount + 1
    Else frame3.Visible = True
    End If

Objectives(1).MyWeight = weight

' add force required to the GJumperSum array and to sum GForce
GJumperSum(1) = Objectives(1).MyObjForce
GForce = GForce + GJumperSum(1)

' Calculate Speed as Required
If Combo1.Text = "Cool-> Below 50°F" Then GTemp = 1
If Combo1.Text = "Mild-> 51°F to 75°" Then GTemp = 2
If Combo1.Text = "Warm-> Above 76°F" Then GTemp = 3
If Combo2.Text = "25 % and less" Then GHumidity = 1
If Combo2.Text = "50 %" Then GHumidity = 2
If Combo2.Text = "75 % and higher" Then GHumidity = 3
If Combo3.Text = "Night" Then GCloud = 3
If Combo3.Text = "Clear Sky" Then GCloud = 1
If Combo3.Text = "Partly_Cloudy" Then GCloud = 4
If Combo3.Text = "Cloudy" Then GCloud = 2

' Unique to each Objective
Dim slope, terrain As Integer
If Combo4.Text = "Level" Then slope = 2
If Combo4.Text = "Slopes Up" Then slope = 3
If Combo4.Text = "Slopes Down" Then slope = 1
If Combo5.Text = "Asphalt" Then terrain = 1
If Combo5.Text = "Hard Dirt" Then terrain = 2
If Combo5.Text = "Loose Sand" Then terrain = 3

Objectives(1).MySpeed = Speed(GTemp, GHumidity, GCloud, slope, terrain)

'compute distances from each position to my objective and places in the
distance matrix

Dim MyX, MyY As Long
MyX = Objectives(1).MyObjX
MyY = Objectives(1).MyObjY

Dim a, b As Integer
b = 1  ' this objectives number in sequence
For a = 1 To GJumpPass * 2
    GDistances(a, 1) = weight * Sqr((GXLoc(a) - MyX) ^ 2 + (GYLoc(a) -
MyY) ^ 2)
    GTime(a, 1) = GDistances(a, 1) / Objectives(1).MySpeed
Next a

' output information to obj****.txt files
    Open "c:\manifest\ObjName.txt" For Output As #20
        Write #20, Objectives(b).MyObjName
    Close #20
    Open "c:\manifest\ObjUnit.txt" For Output As #21
        Write #21, Objectives(b).MyObjUnit
    Close #21
    Open "c:\manifest\ObjX.txt" For Output As #22
        Write #22, Objectives(b).MyObjX
    Close #22
    Open "c:\manifest\ObjY.txt" For Output As #23
        Write #23, Objectives(b).MyObjY
    Close #23
    Open "c:\manifest\ObjForce.txt" For Output As #24
        Write #24, Objectives(b).MyObjForce
    Close #24
    Open "c:\manifest\ObjSpeed.txt" For Output As #25
        Write #25, Objectives(b).MySpeed
    Close #25
```

```
' goto next obj form if required
    If GNumObj > 1 Then
        Obj2.Visible = True
    Else form2.Visible = True
    End If

End Sub

Sub ObjAlt_Change ()
    Objectives(1).MyObjAlt = ObjAlt.Text
End Sub

Sub ObjHelp_Click ()
    frame2.Visible = True
End Sub

Sub ObjName_Change ()

    Objectives(1).MyObjName = ObjName.Text

End Sub

Sub ObjX_Change ()

    Objectives(1).MyObjX = ObjX.Text

End Sub

Sub ObjY_Change ()

    Objectives(1).MyObjY = ObjY.Text

End Sub

Sub ReturnToLast_Click ()
    ObjInfo.Visible = False
End Sub

Sub Unit_Change ()

    Objectives(1).MyObjUnit = Unit.Text

End Sub

Begin Form Obj2 (NOTE : THIS IS INDICITIVE OF FORMS Obj3 thru Obj8)


Sub Form_Load ()
    Combo4.AddItem "Level"                  'grade
    Combo4.AddItem "Slopes Up"
    Combo4.AddItem "Slopes Down"
    Combo5.AddItem "Asphalt"                'terrain
    Combo5.AddItem "Hard Dirt"
    Combo5.AddItem "Loose Sand"
End Sub

Sub NextObj_Click ()

' assign this objectives distance weight coefficient
    Dim weight As Single
    If MsnEssnl.Value = True Then
        weight = GME: GMECount = GMECount + 1
    ElseIf MsnSpt.Value = True Then
        weight = GMS: GMSCount = GMSCount + 1
    ElseIf OSptMsn.Value = True Then
        weight = GOSM: GOSMCount = GOSMCount + 1
    End If

' add values of text boxes to the user defined objective type array

Objectives(2).MyObjName = ObjName.Text
Objectives(2).MyObjUnit = Unit.Text
Objectives(2).MyObjForce = Force.Text
Objectives(2).MyObjX = ObjX.Text
Objectives(2).MyObjY = ObjY.Text
Objectives(2).MyObjAlt = ObjAlt.Text
Objectives(2).MyWeight = weight

' Unique to each Objective;  calculates MySpeed
Dim slope, terrain As Integer
If Combo4.Text = "Level" Then slope = 2
If Combo4.Text = "Slopes Up" Then slope = 3
If Combo4.Text = "Slopes Down" Then slope = 1
If Combo5.Text = "Asphalt" Then terrain = 1
If Combo5.Text = "Hard Dirt" Then terrain = 2
If Combo5.Text = "Loose Sand" Then terrain = 3

Objectives(2).MySpeed = Speed(GTemp, GHumidity, GCloud, slope, terrain)

' add force required to the GJumperSum array
GJumperSum(2) = Objectives(2).MyObjForce
GForce = GForce + GJumperSum(2)

'compute distances from each position to my objective and places in the
distance matrix
Dim MyX, MyY As Long
MyX = Objectives(2).MyObjX
MyY = Objectives(2).MyObjY

Dim NumJumpers As Integer
NumJumpers = GJumpPass * 2

Dim a, b As Integer
b = 2  ' this objectives number in sequence
```

```
For a = 1 To NumJumpers
    GDistances(a, 2) = weight * Sqr((GXLoc(a) - MyX) ^ 2 + (GYLoc(a) -
MyY) ^ 2)
    GTime(a, 2) = GDistances(a, 2) / Objectives(2).MySpeed
Next a

' Output(Append) obj info to the obj****.txt files
    Open "c:\manifest\ObjName.txt" For Append As #20
        Write #20, Objectives(b).MyObjName
    Close #20
    Open "c:\manifest\ObjUnit.txt" For Append As #21
        Write #21, Objectives(b).MyObjUnit
    Close #21
    Open "c:\manifest\ObjX.txt" For Append As #22
        Write #22, Objectives(b).MyObjX
    Close #22
    Open "c:\manifest\ObjY.txt" For Append As #23
        Write #23, Objectives(b).MyObjY
    Close #23
    Open "c:\manifest\ObjForce.txt" For Append As #24
        Write #24, Objectives(b).MyObjForce
    Close #24
    Open "c:\manifest\ObjSpeed.txt" For Append As #25
        Write #25, Objectives(b).MySpeed
    Close #25

' goto next obj form if required
    If GNumObj > 2 Then
        Obj3.Visible = True
    Else form2.Visible = True
    End If


End Sub

Sub ObjX_Change ()
    GObj2X = ObjX.Text
End Sub

Sub ObjY_Change ()
    GObj2Y = ObjY.Text
End Sub

Sub Form_Load ()
' warning code under developement
' do not use other than for this research project
' may (and most definitely does) contain internal bugs
' is the sole property of its developer:
'
'       CPT David D. Briggs

' display message if to many forces are designated
If GForce > GChalks * GNumJumpDoor * 2 Then
```

```
    MsgBox "The total forces as specified exceeds the paratroops
available", 48
End If

' places in the number of null jumpers in the force array (GJumperSum)
GJumperSum(GNumObj + 1) = (GChaulks * GNumJumpDoor * 2) - GForce


' *************************************************************
' FILLS OUT THE REMAINDER OF GTime Array FOR SUBSEQUENT PASSES
' First Pass Information has already been entered

Dim s, t, u, v, w, X, y, count As Integer        ' counters

If GPasses > 1 Then               'Only Multiple Passes
    For w = 2 To GNumJumpDoor \ GJumpPass          ' just the even
passes
        For X = 1 To GJumpPass * 2
            For y = 1 To GNumObj
                GTime(GJumpPass * 2 * (w - 1) + X, y) = GTime(X, y) +
(GRace * 60 * (w - 1))
            Next y
        Next X
    Next w
    t = GNumJumpDoor * 2 Mod (GPasses - 1) * GJumpPass * 2' returns the
number in the uneven pass
    s = (GNumJumpDoor * 2) - t                      ' last even jumper
    For u = 1 To t
          For v = 1 To GNumObj
            GTime(u + s, v) = GTime(u, v) + (GRace * 60 * (GPasses - 1))
          Next v
    Next u
End If

' ***********************************************
' PART I - DEVELOP INITIAL BFS FOR TRANSPO PROBLEM
' ***********************************************
Dim ColumnSum(), RowSum() As Integer
ReDim RowSum(GNumJumpDoor * 2)      ' Sum of Each Row     ->GSupply()
ReDim ColumnSum(GNumObj + 1)        ' Sum of Each Column  ->GJumperSum()
Dim least As Single                 ' floating least position in a column
ReDim GTranspo(GNumJumpDoor * 2, GNumObj + 1, 5)
BFS = 1                             ' Initial Basic Feasible Solution
working = 2                         ' Work in progress
benefit = 3                         ' Benefit Coefficients
updated = 4                         ' Updated Solution
final = 5                           ' Final solution
ReDim GSolution(GNumJumpDoor * 2, GChalks)
Dim MyMark As Integer

' copy the GTime Matrix to a Coeff Matrix
Dim aa, bb  As Integer

ReDim Coeff(GNumJumpDoor * 2, GNumObj + 1)
```

```
For aa = 1 To GNumJumpDoor * 2
    For bb = 1 To GNumObj + 1
        Coeff(aa, bb) = GTime(aa, bb)
    Next bb
Next aa

'*********************************************
'*******   DATA DISPLAY PROCEDURE  ***********
'*********************************************

' set # Cols = Number of Chalks + 4 for posns
grid1.Cols = GChalks + 4

' Fill in Headers on grid
grid1.Row = 0
grid1.ColWidth(0) = 1     ' null
grid1.ColWidth(1) = 400   ' Left/ Right
grid1.ColWidth(2) = 500   ' count
grid1.ColWidth(3) = 500   ' pass

grid1.Col = 1
grid1.Text = "L/R"
grid1.Col = 2
grid1.Text = "Posn"
grid1.Col = 3
grid1.Text = "Pass"

' enters posn and pass data
Dim l, lcount, r, rcount As Integer
For l = 1 To GNumJumpDoor
    grid1.Col = 1
    grid1.Row = l
    grid1.Text = "L"
    grid1.Col = 2
    lcount = lcount + 1
    grid1.Text = lcount
Next l

For r = GNumJumpDoor + 1 To GNumJumpDoor * 2
    grid1.Col = 1
    grid1.Row = r
    grid1.Text = "R"
    grid1.Col = 2
    rcount = rcount + 1
    grid1.Text = rcount
Next r


grid1.ColWidth(4) = 900
grid1.Col = 4
grid1.Row = 0
grid1.Text = "Chalk:1"

Dim a, ac As Integer
```

```
a = 4          'sets off the position and far left columns

For ac = 2 To GChalks
    grid1.Col = ac + 3
    grid1.ColWidth(ac + 3) = 900
    grid1.Text = ac
Next ac



'display which pass on grid1
Dim pass As Integer
grid1.Col = 3
For pass = 1 To GPasses - 1
    For Row = 1 To GJumpPass
        grid1.Row = Row + ((pass - 1) * GJumpPass)
        grid1.Text = pass
    Next Row
    For Row = 1 To GJumpPass
        grid1.Row = Row + GNumJumpDoor + ((pass - 1) * GJumpPass)
        grid1.Text = pass
    Next Row
Next pass

' last pass display procedure
For Row = ((GPasses - 1) * GJumpPass) + 1 To GNumJumpDoor
        grid1.Row = Row
        grid1.Text = GPasses
Next Row
For Row = ((GPasses - 1) * GJumpPass) + 1 To GNumJumpDoor
        grid1.Row = Row + GNumJumpDoor
        grid1.Text = GPasses
Next Row

'grid2   DISPLAY REQ ***************************

Grid2.Cols = GNumObj + 1 + 2 + 1
Grid2.Rows = (GNumJumpDoor * 2) + 3
'headers
Grid2.Row = 0
Grid2.ColWidth(0) = 500
For count = 1 To GNumObj + 3
    Grid2.ColWidth(count) = 500
Next count
'***********************************************
' **** Temporary End of Display Portion *******************



'******************************************
'****** Vogel's Approximation Method ******
'******************************************
'NOTES:
'step 1 tested 10 JAN 95 9:55am
```

```
Dim row, column, MyMark, MyMarkCol, MyMarkC, x As Integer
Dim BigCol, BigRow, RowCol, CrossOut, WhichCross As Integer
Dim penalty() As Single
Dim penaltyC() As Single
Dim least, NextLeast, Biggest, TCost As Single
Dim GTime2() As Single
ReDim GTime2(GNumJumpDoor * 2, GNumObj)
ReDim penalty(GNumObj)
ReDim penaltyC(GNumJumpDoor * 2)
Dim RowCrossOut() As Integer      'keep track of crossed out rows
Dim ColCrossOut() As Integer      'keep track of crossed out columns
ReDim RowCrossOut(GNumJumpDoor * 2)
ReDim ColCrossOut(GNumObj)
Dim iteration, FindCol, FindRow, LastUnCol, LastUnRow As Integer
Dim FoundRow, FoundCol, ColSatisfied As Integer

'resest the coeff array as reqd
For column = 1 To GNumObj
    For row = 1 To GNumJumpDoor * 2
        GTime2(row, column) = GTime(row, column)
    Next row
Next column


'resets the coeff to those available


For column = 1 To GNumObj
    For row = 1 To GNumJumpDoor * 2
        Coeff(row, column) = GTime2(row, column)
    Next row
Next column

 'HERE IS THE LOOP START POINT
ColSatisfied = 0'start with none satisfied
'For iteration = 1 To 19
Do


'Step 1.  Search columns for column penalty between smallest costs



'For column = 1 To GnumObj
column = 0 'reset for this loop
Do
   column = column + 1 'counter
    If ColCrossOut(column) > 0 And column = GNumObj Then
        Exit Do     'bypass if crossed out, exit if last
    ElseIf ColCrossOut(column) > 0 Then
        column = column + 1  'will force skip of this col
    End If
    least = 999999
    NextLeast = 999999
    row = 0
    Do  'find least in col
        row = row + 1
        If RowCrossOut(row) > 0 And row = GNumJumpDoor * 2 Then
            Exit Do   'on the last row, and it has been croossed out
        ElseIf RowCrossOut(row) > 0 Then
            row = row + 1: 'skip that row
        End If
        If Coeff(row, column) < least Then
            MyMark = row: least = Coeff(row, column)
        End If
    Loop Until row = GNumJumpDoor * 2
    Coeff(MyMark, column) = 999999                  'pull from
contention
    row = 0
    Do  'find next least in col
        row = row + 1
        If RowCrossOut(row) > 0 And row = GNumJumpDoor * 2 Then
            Exit Do   'on the last row, and it has been croossed out
        ElseIf RowCrossOut(row) > 0 Then
            row = row + 1'skip that row
        End If
        If Coeff(row, column) < NextLeast Then
            NextLeast = Coeff(row, column)
        End If
    Loop Until row = GNumJumpDoor * 2

    penalty(column) = NextLeast - least
    Coeff(MyMark, column) = GTime2(MyMark, column)   'returns coeff
value !!!

Loop Until column >= GNumObj
'Next column

'which col had greatest penalty      'works 17 MAR
Biggest = 0
For column = 1 To GNumObj
    If penalty(column) > Biggest Then
        Biggest = penalty(column): BigCol = column
    End If
    grid3.row = 0
    grid3.Col = column
    grid3.Text = penalty(column)
Next column

'1a. Search rows for row penalty between smallest costs
'For Row = 1 To GNumJumpDoor * 2
row = 0     'reset for this search
Do
    row = row + 1
    If RowCrossOut(row) > 0 And row = GNumJumpDoor * 2 Then
        Exit Do
    ElseIf RowCrossOut(row) > 0 Then
        row = row + 1'skip this row
    End If

    least = 999999
    NextLeast = 999999
    column = 0
    Do
        column = column + 1 'column counter
        If ColCrossOut(column) > 0 And column = GNumObj Then
            Exit Do
        ElseIf ColCrossOut(column) > 0 Then
            column = column + 1 'skip that column
        End If
        If Coeff(row, column) < least Then
            MyMarkC = column: least = Coeff(row, column)
        End If
    Loop Until column >= GNumObj
    Coeff(row, MyMarkC) = 999999         'pull from contention

    column = 0           'reset counter
    Do             'find next least in col
        column = column + 1 'column counter
        If ColCrossOut(column) > 0 And column = GNumObj Then
            Exit Do
        ElseIf ColCrossOut(column) > 0 Then
            column = column + 1 'skip that column
        End If
        If Coeff(row, column) < NextLeast Then
            NextLeast = Coeff(row, column)
        End If
    Loop Until column >= GNumObj
    '@@@@@@@@@@@@@@@@@@
    If RowCrossOut(row) > 0 Then
        penaltyC(row) = 0
    Else
        penaltyC(row) = NextLeast - least
    End If
    '@@@@@@@@@@@@@@@@@@
    Coeff(row, MyMarkC) = GTime2(row, MyMarkC) ' reset the value !!!!!
'Next Row
Loop Until row >= GNumJumpDoor * 2

'which row had greatest penalty       'works 17 MAR
Biggest = 0
grid3.Col = 0
row = 0'reset for this search
Do
    row = row + 1
    If RowCrossOut(row) = 0 Then LastUnRow = row 'keep track of last
uncrossed row
    If RowCrossOut(row) > 0 And row = GNumJumpDoor * 2 Then
        Exit Do
    ElseIf RowCrossOut(row) > 0 Then 'find next empty row
```

```
        For FindRow = row + 1 To GNumJumpDoor * 2
            If RowCrossOut(FindRow) = 0 Then
                row = FindRow: FoundRow = 1: Exit For
            End If
        Next FindRow                     'IMPLIMENT FOR COL ALSO !
    End If                           '**************************


    grid3.row = row
    grid3.Text = penaltyC(row)
    If penaltyC(row) > Biggest Then
        Biggest = penaltyC(row): BigRow = row
    End If
Loop Until row >= GNumJumpDoor * 2



'find biggest penalty between rows and column
Print penalty(BigCol), penaltyC(BigRow)
If penalty(BigCol) > penaltyC(BigRow) Then
    CrossOut = BigCol: RowCol = 10 'will allocate to col
Else CrossOut = BigRow: RowCol = 20  'will allocate to row
End If

'find smallest cost in that cross out line and allocate all to it
Select Case RowCol
Case 10      'will allocate to this col

    least = 999999      'find smallest value in that col
    row = 0
    Do
        row = row + 1
        If RowCrossOut(row) > 0 And row = GNumJumpDoor * 2 Then
            Exit Do  'MAY NOT NEED THIS HERE
        ElseIf RowCrossOut(row) > 0 Then
            row = row + 1: Print 'skip this row
        End If
        If Coeff(row, CrossOut) < least Then
            least = Coeff(row, CrossOut): MyMarkRow = row
        End If
    Loop Until row >= GNumJumpDoor * 2
    'allocate everything to it
    reqs = GJumperSum(CrossOut) - ColumnSum(CrossOut)
    seats = GSupply(MyMarkRow) - RowSum(MyMarkRow)
    allocation = 0
    If reqs And seats > 0 Then
        If seats > reqs Then seats = reqs:
        GTranspo(MyMarkRow, CrossOut, 1) = seats:
        allocation = 1
    Else GTranspo(MyMarkRow, CrossOut, 1) = 0     'assign basic value
    End If

   'bookkeeping
```

```
    If allocation = 1 Then           'update sums
        ColumnSum(CrossOut) = ColumnSum(CrossOut) + GTranspo(MyMarkRow,
CrossOut, 1)
        RowSum(MyMarkRow) = RowSum(MyMarkRow) + GTranspo(MyMarkRow,
CrossOut, 1)
    End If

    'Cross out column or row
    If GJumperSum(CrossOut) = ColumnSum(CrossOut) Then   'cross col if
full
        WhichCross = 1
    Else WhichCross = 2  'cross row out other wise (nulls will take up
slack)
    End If
    Select Case WhichCross
    Case 1     'cross out column
        ColCrossOut(CrossOut) = 1
        ColSatisfied = ColSatisfied + 1

    Case 2     'cross out row
        RowCrossOut(MyMarkRow) = 1
    End Select

Case 20                             'allocates to a row
    least = 999999
    column = 0  'reset
    Do
        column = column + 1
        If ColCrossOut(column) > 0 And column = GNumObj Then
            Exit Do: Print "exit case 20 part 1"
        ElseIf ColCrossOut(column) > 0 Then
            column = column + 1: Print "exit case 20 part 2" 'skip that
column
        End If
        If Coeff(CrossOut, column) < least Then
            least = Coeff(CrossOut, column): MyMarkCol = column
        End If
    Loop Until column >= GNumObj
    'allocate everything to it
    reqs = GJumperSum(MyMarkCol) - ColumnSum(MyMarkCol)
    seats = GSupply(CrossOut) - RowSum(CrossOut)
    allocation = 0
    If reqs And seats > 0 Then
        If seats > reqs Then seats = reqs
        GTranspo(CrossOut, MyMarkCol, 1) = seats
        allocation = 1
    Else GTranspo(CrossOut, MyMarkCol, 1) = 0
    End If
    'bookkeeping
    If allocation = 1 Then           'update sums
        ColumnSum(MyMarkCol) = ColumnSum(MyMarkCol) + GTranspo(CrossOut,
MyMarkCol, 1)
        RowSum(CrossOut) = RowSum(CrossOut) + GTranspo(CrossOut,
MyMarkCol, 1)
```

```
    End If

    'Cross out column or row
    If GJumperSum(MyMarkCol) = ColumnSum(MyMarkCol) Then   'cross col if
full
        WhichCross = 1
    Else WhichCross = 2  'cross row out otherwise (nulls will take up
slack)
    End If
    Select Case WhichCross
    Case 1     'cross out column
          ColCrossOut(MyMarkCol) = 1
          ColSatisfied = ColSatisfied + 1
    Case 2     'cross out row
          RowCrossOut(CrossOut) = 1
    End Select
End Select

' reset all temporary values
BigCol = 0
BigRow = 0
MyMarkRow = 0
MyMarkCol = 0
CrossOut = 0
For column = 1 To GNumObj
    penalty(column) = 0
Next column
For row = 1 To GNumJumpDoor * 2
    penaltyC(row) = 0
Next row

For column = 1 To GNumObj
    grid1.Col = column
    For row = 1 To GNumJumpDoor * 2
    grid1.row = row
    grid1.Text = Coeff(row, column)
    Next row
Next column

' LOOP FROM THIS POINT BACK, m+n-1 times
Loop Until ColSatisfied = GNumObj
'Print in grid3 the biggest penalty and where it Is (Col)

'compute associated transportation cost for BFS
'   and place in a file
For Row = 1 To GNumJumpDoor * 2
    For column = 1 To GNumObj + 1
        TransCost = TransCost + (Coeff(Row, column) * GTranspo(Row,
column, 1))
    Next column
Next Row

Open "c:\manifest\Trancost.txt" For Output As #1
    Write #1, TransCost
```

```
Close #1


'display manifest matrix after computation done by LoadPlan.FRM
' of GTranspo(row, column, 1)
'must search row by row for non-zero values,
' mark the location, and translate into display locations

Dim position, counter, chalk   As Integer
ReDim GSolution(GChalks, GNumJumpDoor * 2)

For Row = 1 To GNumJumpDoor * 2
    counter = 1 'resets the value at the 1st column

    For column = 1 To GNumObj + 1
        If GTranspo(Row, column, 1) <> 0 Then

            For chalk = counter To GTranspo(Row, column, 1)
                GSolution(chalk, Row) = Objectives(column).MyObjUnit
            Next chalk

            counter = GTranspo(Row, column, 1) + 1

        End If
    Next column
Next Row

'**** display the solution in grid1 ***************

For Row = 1 To GNumJumpDoor * 2 Step 2   'displays all left door
    For column = 1 To GChalks
        grid1.Col = column + 3
        grid1.Row = (Row + 1) / 2
        grid1.Text = GSolution(column, Row)
    Next column
Next Row


For Row = 2 To GNumJumpDoor * 2 Step 2   'displays all right door
   For column = 1 To GChalks
        grid1.Col = column + 3
        grid1.Row = GNumJumpDoor + (Row / 2)
        grid1.Text = GSolution(column, Row)
   Next column
Next Row

'output to the left.txt file to be read in MODSIM II simulation
Open "c:\manifest\left.txt" For Output As #5
For column = 1 To GChalks
    grid1.Col = column + 3
    For Row = 1 To GNumJumpDoor
        grid1.Row = Row
        Write #5, grid1.Text
    Next Row
Next column
Close #5


'output to the right.txt file to be read in MODSIM II simulation
Open "c:\manifest\right.txt" For Output As #6
For column = 1 To GChalks
    grid1.Col = column + 3
    For Row = GNumJumpDoor + 1 To GNumJumpDoor * 2
        grid1.Row = Row
        Write #6, grid1.Text
    Next Row
Next column
Close #6

End Sub

Sub Command8_Click ()
'LAUNCH SIMULATION

Dim X
X = Shell("c:\manifest\simulatn.exe", 1)

' this works !!
End Sub
```

APPENDIX B

MODSIM II SIMULATION CODE

```
 DEFINITION MODULE abn;

{ Contains modules and objects for use in
  Mass Tactical Airborne Simulation Project,
  as part of thesis conceived by David D. Briggs

 ORIGINAL PROGRAM DATE: 10 FEB 95

 LAST REVISION: 25 MAR 95 1100

 CHANGES MADE ON LAST VERSION: verify aircraft location  X
                               verify distance calculation  X
                                add report collector object  X
                                 multiple replications  X
                                  Final report format  X
                                      multiple passes
                                       wind shift   X
                                        Output jumper times to array  X
PREVIOUS LATEST VERSION REVISION DATE: 23 MAR 95
}

FROM SimMod      IMPORT StartSimulation, SimTime;
FROM RandMod     IMPORT RandomObj, FetchSeed;
FROM GrpMod      IMPORT QueueObj;
FROM StatMod     IMPORT SREAL, RStatObj;
FROM IOMod       IMPORT StreamObj, FileUseType(Input), FileUseType(Output);
FROM GrpMod      IMPORT QueueObj;
FROM MathMod     IMPORT SIN, COS;

TYPE
        ReadFrom = OBJECT {makes all input values global}
         LPasses, LChalks, LJumpPass, LRace,
         LNumObj, LNumJumpDoor,
         StringCount, RealCount, IntCount    :INTEGER;
         UDoF, UACSpeed, LWindDirDelta, LWindSpeedDelta, UWindSpeedDelta,
UWindDirDelta,
         LDoF, LPPIX, LPPIY, LACSpeed, LInterval        :REAL;
         LObjName, LObjUnit        :STRING;
         ASK METHOD InitGlobalINT(IN LPasses, LChalks, LJumpPass, LRace,
LNumObj, LNumJumpDoor :INTEGER);
         ASK METHOD InitGlobalREAL(IN LDoF, LPPIX, LPPIY, LACSpeed,
LInterval, LWindSpeedDelta, LWindDirDelta :REAL);
         ASK METHOD InitArrays;
         ASK METHOD InitObjectives;
         ASK METHOD InitArrayString(IN LObjName, LObjUnit          :STRING);
        ASK METHOD InitArrayReal(IN LObjX, LObjY, LObjSpeed    :REAL);
        ASK METHOD InitArrayInt(IN LObjForce :INTEGER);
        ASK METHOD PrintGlobals;
        ASK METHOD ReadManifest;
        ASK METHOD UpDateNextPass(IN UDoF, UACSpeed, UWindSpeedDelta,
UWindDirDelta :REAL);
       END OBJECT;


       ParatrooperObj = OBJECT
        MyX, MyY, MyXLoc, MyYLoc, MySpeed, MyFallTime, SpeedVar,
        DoorToObjTime : REAL;
        Chalk, MyObjNum   : INTEGER;
        MyUnit         : STRING;
        ASK METHOD ObjInit;
        ASK METHOD IDSelf(IN Unit :STRING);
        TELL METHOD Jump (IN MyX, MyY, MyFallTime : REAL);
        ASK METHOD FindMyObj;
        TELL METHOD GotoMyObj;
       END OBJECT;        {ParatrooperObj}


       ObjectiveObj = OBJECT (RStatObj)
        MyObjX, MyObjY, {MyObjSpeed}
        ArrivalTime, ShowTime : REAL;
        hundred, ninety,
        seventyfive, fifty      : SREAL;
        JumpCounter, report : INTEGER;
        AssignedJumpers, ObjNum, RunningCounter  : INTEGER;
        MyName, MyUnit    : STRING;
        MyArrivalTimes      : ARRAY INTEGER OF REAL;
        ASK METHOD ObjInit;
        ASK METHOD MakeAssignments(IN MyNum     :INTEGER);
        ASK METHOD CountJumper(IN ShowTime :REAL);
        ASK METHOD ResetObjective;
        ASK METHOD PrintArrivalTimes;
       END OBJECT;        {ObjectiveObj}


       AircraftObj = OBJECT (QueueObj)
        CurrentX, CurrentY, ACStartX, ACStartY,
        CurrentAlt, NextAC            : REAL;
        NumberOfJumpers,
        Chalk, ChalkNum, PassNum           : INTEGER;
        PassTime,MeterSec  : REAL;
```

```
          ASK METHOD ObjInit;
          ASK METHOD StartUp(IN ACStartX, ACStartY :REAL);   {initializes the start
point}
          {TELL METHOD Fly;        moves AC, calcs new x y, for interval sec.}

          TELL METHOD DropEmL(IN ChalkNum :INTEGER);   {causes jumper to
exit, update locations}
           {must calc departure time, x and y's}
          OVERRIDE
          ASK METHOD Remove() :ANYOBJ;
        END OBJECT;          {AircraftObj}


        RightDoor = OBJECT (AircraftObj)
         LROffSet      :REAL;{places buffer to start after leftdoor}
         LRDelay       :REAL;
           ASK METHOD StartUpR(IN ACStartX, ACStartY, LROffSet :REAL);
           TELL METHOD DropEmR(IN ChalkNum :INTEGER);
         END OBJECT;

        ReportCollect = OBJECT
           ASK METHOD ObjInit;
          ASK METHOD RepReport(IN repNum :INTEGER);
          ASK METHOD FinalReport;
        END OBJECT;




        VAR
         CARPXCoord, CARPYCoord,
         JumpAltitude, acNum,
         x, m              :INTEGER;
         Heading, DoF, WindSpeedDelta, WindDirDelta,
         PPIX, PPIY, Serial, RSerial                :REAL;
         ACType, Passes, X, Y, Z, OCount, JCount, LCount, RCount,
         Chalks, JumpPass, repNum,
         Race, NumObj,
         NumJumpDoor   :INTEGER;
         NullString    :STRING;
         LeftJump,
         RightJump     :ARRAY INTEGER, INTEGER OF STRING;
         ObjName       :ARRAY INTEGER OF STRING;
         ObjUnit       :ARRAY INTEGER OF STRING;
         ObjX          :ARRAY INTEGER OF REAL;
```

```
          ObjY          :ARRAY INTEGER OF REAL;
          ObjForce      :ARRAY INTEGER OF INTEGER;
          ObjSpeed      :ARRAY INTEGER OF REAL;
          MyObj                 :ARRAY INTEGER OF ObjectiveObj;
          LeftDoorObj :ARRAY INTEGER OF AircraftObj;
          RightDoorObj          :ARRAY INTEGER OF RightDoor;
          FIFTY                 :ARRAY INTEGER OF SREAL;
          SEVENTYFIVE           :ARRAY INTEGER OF SREAL;
          NINETY        :ARRAY INTEGER OF SREAL;
          HUNDRED  :ARRAY INTEGER OF SREAL;
          ACSpeed, Interval, Altitude,
          DoorInterval :REAL;
          Rand, ACRand, ACRand2, RandJ, RandO  :RandomObj;
          Jumper, JUMPER :ParatrooperObj;
          OutStrm       :StreamObj;

END MODULE;
```

```
IMPLEMENTATION MODULE abn;
{See and ensure updating of notes in the definition Module abn

NOTES: Verified aircraft and jumper location updates 13 MAR 95
       Verified distance and wait times 13 MAR 95


LAST MODIFICATION DATE: 15 MAR 95
LOCATION: ISTS

PREVIOUS VERSION DATE 28 FEB 95
}

FROM SimMod      IMPORT StartSimulation, SimTime;
FROM RandMod     IMPORT RandomObj, FetchSeed;
FROM GrpMod      IMPORT QueueObj;
FROM StatMod     IMPORT SREAL, RStatObj;
FROM IOMod       IMPORT StreamObj, FileUseType(Input), FileUseType(Output);
FROM MathMod     IMPORT SQRT, COS, SIN;
FROM GrpMod      IMPORT QueueObj;


OBJECT ReadFrom;  {makes all input values global}

 ASK METHOD InitGlobalINT(IN LPasses, LChalks, LJumpPass, LRace, LNumObj,
LNumJumpDoor :INTEGER);
     VAR
      EmptyString :STRING;
     BEGIN
      OUTPUT("Reading Input Parameters........");
      Passes:=LPasses;
      Chalks:=LChalks;
      Race:=LRace;
      NumObj:=LNumObj;
      NumJumpDoor:=LNumJumpDoor;
      JumpPass:= LJumpPass;
      NEW(OutStrm);
     END METHOD;

 ASK METHOD InitGlobalREAL(IN LDoF, LPPIX, LPPIY, LACSpeed, LInterval,
LWindSpeedDelta, LWindDirDelta :REAL);
     BEGIN
      DoF:=LDoF;
      PPIX:=LPPIX;
      PPIY:=LPPIY;
     ACSpeed:=LACSpeed;
     Interval:=LInterval;
     WindSpeedDelta:=LWindSpeedDelta;
     WindDirDelta:=LWindDirDelta;
    END METHOD;
ASK METHOD InitArrays;
    BEGIN
     NEW(ObjName, 1..NumObj);
     NEW(ObjUnit, 1..NumObj);
     NEW(ObjX, 1..NumObj);
     NEW(ObjY, 1..NumObj);
     NEW(ObjForce, 1..NumObj);
     NEW(ObjSpeed, 1..NumObj);
     NEW(LeftJump, 1..NumJumpDoor, 1.. Chalks);
     NEW(RightJump, 1..NumJumpDoor, 1.. Chalks);
    END METHOD;
ASK METHOD InitArrayString(IN LObjName, LObjUnit         :STRING);
    BEGIN
     INC(StringCount);
     ObjName[StringCount]:=LObjName;
     ObjUnit[StringCount]:=LObjUnit;
    END METHOD;
ASK METHOD InitArrayReal(IN LObjX, LObjY, LObjSpeed    :REAL);
    BEGIN
     INC(RealCount);
     ObjX[RealCount]:=LObjX;
     ObjY[RealCount]:=LObjY;
     ObjSpeed[RealCount]:=LObjSpeed;
    END METHOD;
ASK METHOD InitArrayInt(IN LObjForce :INTEGER);
    BEGIN
     INC(IntCount);
     ObjForce[IntCount]:=LObjForce;
    END METHOD;
ASK METHOD InitObjectives;
    VAR
     count :INTEGER;
    BEGIN
    NEW(MyObj, 1..NumObj);
     FOR count := 1 TO NumObj
      NEW(MyObj[count]);
      ASK MyObj[count] TO MakeAssignments(count);
     END FOR;
    END METHOD;
```

```
  ASK METHOD PrintGlobals;  {*TEMP*}
        BEGIN
        {OUTPUT("**********************************");
        OUTPUT(" # Chalks = ",Chalks);
        OUTPUT(" # Passes = ",Passes);
        OUTPUT(" # OBJ    = ",NumObj);
        OUTPUT(" PPIX, PPIY", PPIX, " ", PPIY);
        OUTPUT(" SPEED, DoF  = ", ACSpeed, " ", DoF);
        OUTPUT("**********************************");
        OUTPUT;}
        END METHOD;
  ASK METHOD UpDateNextPass(IN UDoF, UACSpeed, UWindSpeedDelta,
UWindDirDelta         :REAL);
        BEGIN
         ACSpeed:=UACSpeed;
         DoF:=UDoF;
         WindSpeedDelta:=UWindSpeedDelta;
         WindDirDelta:=UWindDirDelta;
         OUTPUT;
         {OUTPUT("New serial speed and direction ", ACSpeed," Knots ", DoF, "
degrees");
         OUTPUT;}
        END METHOD;
  ASK METHOD ReadManifest;
        VAR
         C, J   :INTEGER;
         Strm2          :StreamObj;
        BEGIN
         NEW(Strm2);
         ASK Strm2 TO Open("Left.txt", Input);
         FOR C:=1 TO Chalks
          FOR J:=1 TO NumJumpDoor
           ASK Strm2 TO ReadLine(LeftJump[J,C]);
          END FOR;
         END FOR;
         ASK Strm2 TO Close;
         DISPOSE(Strm2);

         NEW(Strm2);
         ASK Strm2 TO Open("Right.txt", Input);
         FOR C:=1 TO Chalks
          FOR J:=1 TO NumJumpDoor
           ASK Strm2 TO ReadLine(RightJump[J,C]);
          END FOR;
         END FOR;
```

```
            ASK Strm2 TO Close;
            DISPOSE(Strm2);
           END METHOD;
END OBJECT;




OBJECT ParatrooperObj;
 ASK METHOD ObjInit;
        BEGIN
          INC(JCount);
          NEW(RandJ);
        END METHOD;

 ASK METHOD IDSelf(IN Unit :STRING);
        BEGIN
                MyUnit := Unit;
        END METHOD;          {reads its unit from the generator}

 TELL METHOD Jump(IN MyX, MyY, MyFallTime :REAL); {include drift etc.}
     VAR
          DeRig, TheTime :REAL;
        BEGIN
        MyXLoc:=MyX;
        MyYLoc:=MyY;
        TheTime:=SimTime();
        WAIT DURATION MyFallTime;
        END WAIT;
        DeRig:= ASK RandJ Normal(450.0, 108.0); {Verified}
        WAIT DURATION DeRig;
        END WAIT;
        ASK SELF TO FindMyObj;
        TELL SELF TO GotoMyObj;
        END METHOD;

 ASK METHOD FindMyObj;          {Also computes DoorToObjTime}
        VAR
          XandY, XDist, YDist, MyObjDistance       :REAL;
        BEGIN
        FOR x:=1 TO NumObj   {Determine which Objective to go to}
                IF MyUnit = ObjUnit[x]
                 m := x;
                END IF;
        END FOR;
```

```
         IF STRLEN(MyUnit) <= 1 {clear out the nulls immediately}
          DISPOSE(SELF);
          {OUTPUT("Null Jumper Disposed of ... ");}
         END IF;

         MyObjNum:=m; {solidify which obj in the array}
         XDist:=MyXLoc-ObjX[m];
         YDist:=MyXLoc-ObjY[m];

         SpeedVar:= ASK RandJ Triangular(0.1, 0.75, 0.99);
         XandY:=(MyXLoc-ObjX[m])*(MyXLoc-ObjX[m])+(MyYLoc-
ObjY[m])*(MyYLoc-ObjY[m]);
         MyObjDistance:=SQRT(XandY);
         DoorToObjTime:= MyObjDistance /(ObjSpeed[m]*SpeedVar);
         END METHOD;

 TELL METHOD GotoMyObj;
         VAR
          ShowTime    :REAL;
         BEGIN
         WAIT DURATION DoorToObjTime; {verified passed 13 MAR 95}
         END WAIT;
         ShowTime:=SimTime();
         ASK MyObj[MyObjNum] TO CountJumper(ShowTime);
         DISPOSE(SELF);
         END METHOD;

END OBJECT;



OBJECT ObjectiveObj;
 ASK METHOD ObjInit;
         BEGIN
          INC(OCount);
          NEW(RandO);
          report:=1;
         END METHOD;
 ASK METHOD MakeAssignments(IN MyNum     :INTEGER);
         BEGIN
          ObjNum := MyNum;
          AssignedJumpers:=ObjForce[MyNum];
          MyName:=ObjName[MyNum];
          MyUnit:=ObjName[MyNum];
         NEW(MyArrivalTimes, 1..AssignedJumpers*10); {# in * replications}
        END METHOD;

    ASK METHOD CountJumper(IN ShowTime :REAL);
        VAR
          CurrentPercentage, JCounter, AJumpers    :REAL;
        BEGIN                   {includes the individual derig var here}
        ArrivalTime:= ShowTime;
        INC(JumpCounter);
        INC(RunningCounter);
        MyArrivalTimes[RunningCounter]:=ArrivalTime/60.0;
        JCounter:=FLOAT(JumpCounter);
        AJumpers:=FLOAT(AssignedJumpers);
        CurrentPercentage:=JCounter/AJumpers;
        {OUTPUT("relative combat power = ", CurrentPercentage);
        OUTPUT;}
        CASE report
          WHEN 1:
                IF CurrentPercentage >= 0.50
                 fifty:=SimTime();
                 {OUTPUT("FIFTY PERCENT AT ", MyName, " AT TIME ", fifty);}
                 FIFTY[ObjNum]:=fifty;
                 INC(report);
                END IF;
          WHEN 2:
                IF CurrentPercentage >= 0.75
                 seventyfive:=SimTime();
                 {OUTPUT("SEVENTY-FIVE PERCENT AT ", MyName, " AT TIME
", seventyfive);}
                 SEVENTYFIVE[ObjNum]:=seventyfive;
                 INC(report);
                END IF;
         WHEN 3:
                IF CurrentPercentage >= 0.90
                 ninety:=SimTime();
                 {OUTPUT("NINETY PERCENT AT ", MyName, " AT TIME ",
ninety);}
                 NINETY[ObjNum]:=ninety;
                 INC(report);
                END IF;
         WHEN 4:
                IF CurrentPercentage = 1.0
                 hundred:=SimTime();
                 HUNDRED[ObjNum]:=hundred;
```

```
           {OUTPUT("HUNDRED PERCENT AT ", MyName, " AT TIME ",
hundred);}
          END IF;
     END CASE;
     END METHOD;

 ASK METHOD ResetObjective;
     BEGIN
      report:=1;
      JumpCounter:=0;
      fifty:=0.0;
      seventyfive:=0.0;
      ninety:=0.0;
      hundred:=0.0;
     END METHOD;

     ASK METHOD PrintArrivalTimes;  {done after all stats}
     VAR
      count :INTEGER;
     BEGIN
      ASK OutStrm TO Open(MyName, Output);
      ASK OutStrm TO WriteString(MyName);
       ASK OutStrm TO WriteString(";");
      FOR count:=1 TO AssignedJumpers*7
          ASK OutStrm TO WriteReal(MyArrivalTimes[count], 6, 3);
        ASK OutStrm TO WriteString(";");
      END FOR;
      ASK OutStrm TO Close;
     END METHOD;

END OBJECT;

OBJECT AircraftObj;
 ASK METHOD ObjInit;
     BEGIN
      NEW(ACRand);
      NEW(ACRand2);
     END METHOD;

 ASK METHOD StartUp(IN ACStartX, ACStartY :REAL);
     VAR
      R1, R2 :REAL;   {*****}
     BEGIN
      INC(acNum);
      CurrentX:= PPIX + ACStartX;
      CurrentY:= PPIY + ACStartY;
     END METHOD;

 TELL METHOD DropEmL(IN ChalkNum :INTEGER);
     VAR
      RandNumGen       : RandomObj;
      TimeToImpact     : REAL;
      DriftX, DriftY, TempX, TempY, Interval, WindX, WindY, WindDriftDist,
      Drift1, FallTime,
      JumperX, JumperY,
      DeploymentAlt, Temp1, Temp2, Temp3, Temp4, Temp5, Temp6:REAL;
      J      :INTEGER;
      unit   :STRING;
     BEGIN
        NextAC:=ASK ACRand2 Normal(8.45, 1.4);
        IF ChalkNum > 0
          Serial:=Serial+NextAC;
          WAIT DURATION Serial;   {Time between A/C in a serial}
         END WAIT;
        END IF;
       FOR J:= 1 TO JumpPass
          NEW(Jumper);
          unit:= LeftJump[J,ChalkNum];
          ASK Jumper TO IDSelf(unit);
          DeploymentAlt:= ASK ACRand Normal(679.3, 23.49);
          Drift1:=(DeploymentAlt/100.0)*3.841;
          DriftX:=ASK ACRand Triangular(-Drift1, 0.0, Drift1);
          DriftY:=ASK ACRand Triangular(-Drift1, 0.0, Drift1);
          FallTime:=ASK ACRand Normal(39.2, 4.9);
          WindDriftDist:=FallTime*(WindSpeedDelta*0.5141);
          WindX:=WindDriftDist*SIN(WindDirDelta*0.01745);
          WindY:=WindDriftDist*COS(WindDirDelta*0.01745);
          JumperX:=CurrentX+DriftX+WindX;
          JumperY:=CurrentY+DriftY+WindY;
          {OUTPUT(JumperX, " ", JumperY, " ", FallTime);}
          TELL Jumper TO Jump(JumperX, JumperY, FallTime);
      {FLY for interval seconds}
          Interval:=ASK ACRand Gamma(1.040956, 11.04); {time between
jumpers}
          MeterSec:=ACSpeed * 0.5141;  {convert speed to m/s}

          CurrentX:=CurrentX + Interval*MeterSec*SIN(DoF*0.01745);
          CurrentY:=CurrentY + Interval*MeterSec*COS(DoF*0.01745);
          {OUTPUT("LEFT jumper location  ", JumperX, " ", JumperY, " ",
FallTime); @@@}
```

```
                WAIT DURATION Interval;
                END WAIT;
           END FOR;
           INC(PassNum);
           IF ChalkNum = Chalks   {Reset for each pass/replication}
                Serial := 0.0;
           END IF;
          END METHOD;



          ASK METHOD Remove() : ANYOBJ;
           BEGIN
            JUMPER:= INHERITED Remove();
            RETURN JUMPER;
          END METHOD {Remove};

END OBJECT;

OBJECT RightDoor;

 ASK METHOD StartUpR(IN ACStartX, ACStartY, LROffSet  :REAL);
          VAR
           AdjX, AdjY, TravelDist       :REAL;
          BEGIN
          LRDelay         := LROffSet;
          TravelDist:= 125.0 * 0.5141 * LROffSet;
          AdjX:=SIN(DoF*0.01745)*TravelDist;
          AdjY:=COS(DoF*0.01745)*TravelDist;
          CurrentX:=PPIX + ACStartX + AdjX;
          CurrentY:=PPIY + ACStartY + AdjY;
          {OUTPUT("Adjusted impact Right Door: ", CurrentX, " ", CurrentY);}
 END METHOD;



 TELL METHOD DropEmR(IN ChalkNum :INTEGER);
          VAR
           RandNumGen          : RandomObj;
           TimeToImpact        : REAL;
           DriftX, DriftY, TempX, TempY, Interval,
           Drift1, FallTime, WindX, WindY, WindDriftDist,
           JumperX, JumperY,
           DeploymentAlt :REAL;
           J       :INTEGER;
          unit   :STRING;
        BEGIN

          WAIT DURATION LRDelay;{delay in time}
          END WAIT;
            {OUTPUT("right door Waited duration ",SimTime()," ", LRDelay);}
          NextAC:=ASK ACRand2 Normal(8.45, 1.4);
              IF ChalkNum > 0
                  RSerial:=RSerial+NextAC;
                  WAIT DURATION RSerial;    {Time between A/C in a serial}
                END WAIT;
              END IF;
          FOR J:= 1 TO JumpPass

                  NEW(Jumper);
                  unit:= RightJump[J,ChalkNum];
                  ASK Jumper TO IDSelf(unit);
                  DeploymentAlt:= ASK ACRand Normal(679.3, 23.49);
                  Drift1:=(DeploymentAlt/100.0)*3.841;
                  DriftX:=ASK ACRand Triangular(-Drift1, 0.0, Drift1);
                  DriftY:=ASK ACRand Triangular(-Drift1, 0.0, Drift1);
                  FallTime:=ASK ACRand Normal(39.2, 4.9);
                  WindDriftDist:=FallTime*(WindSpeedDelta*0.5141);
                  WindX:=WindDriftDist*SIN(WindDirDelta*0.01745);
                  WindY:=WindDriftDist*COS(WindDirDelta*0.01745);
                  JumperX:=CurrentX+DriftX+WindX;
                  JumperY:=CurrentY+DriftY+WindY;
                  {OUTPUT("  RIGHT jumper location  ", JumperX, " ", JumperY, " ",
FallTime);
                  TELL Jumper TO Jump(JumperX, JumperY, FallTime);
          {FLY for interval seconds}
                  Interval:=ASK ACRand Gamma(1.040956, 11.04); {time between
jumpers}
                  MeterSec:=ACSpeed * 0.5141;  {convert speed to m/s}

                  CurrentX:=CurrentX + Interval*MeterSec*SIN(DoF*0.01745);
                  CurrentY:=CurrentY + Interval*MeterSec*COS(DoF*0.01745);
                  {OUTPUT("RIGHT Flying New CurrentX: ",CurrentX, " Y: ",
CurrentY); }
                  WAIT DURATION Interval;
                  END WAIT;
         END FOR;
         INC(PassNum);
         IF ChalkNum = Chalks   {Reset for each pass/replication}
                  RSerial := 0.0;
        END IF;
  END METHOD;

END OBJECT;

OBJECT ReportCollect;
  ASK METHOD ObjInit; {size the global collection arrays}
   BEGIN
        NEW(FIFTY, 1..NumObj);
        NEW(SEVENTYFIVE, 1..NumObj);
        NEW(NINETY, 1..NumObj);
        NEW(HUNDRED, 1..NumObj);
  END METHOD;

  ASK METHOD RepReport(IN repNum :INTEGER);
    BEGIN
        OUTPUT("****************************************");
        OUTPUT("***** REPLICATION # ", repNum, " *******************");
        OUTPUT("****************************************");
        OUTPUT;
        FOR X:= 1 TO NumObj
          OUTPUT("      ",ObjName[X], " ", ObjUnit[X]);
          OUTPUT("----------------------------------------");
          OUTPUT("Fifty % at        ", FIFTY[X]/60.0, " minutes");
          OUTPUT("Seventy-Five % at ", SEVENTYFIVE[X]/60.0, " minutes");
          OUTPUT("Ninety % at       ", NINETY[X]/60.0, " minutes");
          OUTPUT("One Hundred % at  ", HUNDRED[X]/60.0, " minutes");
          OUTPUT;
          ASK MyObj[X] TO ResetObjective; {Resets the objective data}
        END FOR;

        OUTPUT("****************************************");
        OUTPUT;
  END METHOD;

  ASK METHOD FinalReport;
        VAR
         mean50, min50, max50, sd50,
         mean75, min75, max75, sd75,
         mean90, min90, max90, sd90,
         mean100, min100, max100, sd100  :REAL;
        BEGIN
        OUTPUT;
        OUTPUT;
        OUTPUT("----------------------------------------");
        OUTPUT("<<<<< SIMULATION SUMMARY REPORT >>>>>>>");
        OUTPUT("----------------------------------------");
        OUTPUT("     (All Statistics in minutes)       ");
        OUTPUT;

        FOR X:= 1 TO NumObj
         mean50:= ASK (GETMONITOR (FIFTY[X], RStatObj)) Mean;
         min50:= ASK (GETMONITOR (FIFTY[X], RStatObj)) Minimum;
         max50:=  ASK (GETMONITOR (FIFTY[X], RStatObj)) Maximum;
         sd50:=  ASK (GETMONITOR (FIFTY[X], RStatObj)) StdDev;
         mean75:= ASK (GETMONITOR (SEVENTYFIVE[X], RStatObj)) Mean;
         min75:= ASK (GETMONITOR (SEVENTYFIVE[X], RStatObj)) Minimum;
         max75:=  ASK (GETMONITOR (SEVENTYFIVE[X], RStatObj)) Maximum;
         sd75:=  ASK (GETMONITOR (SEVENTYFIVE[X], RStatObj)) StdDev;
         mean90:= ASK (GETMONITOR (NINETY[X], RStatObj)) Mean;
         min90:= ASK (GETMONITOR (NINETY[X], RStatObj)) Minimum;
         max90:=  ASK (GETMONITOR (NINETY[X], RStatObj)) Maximum;
         sd90:=  ASK (GETMONITOR (NINETY[X], RStatObj)) StdDev;
         mean100:= ASK (GETMONITOR (HUNDRED[X], RStatObj)) Mean;
         min100:= ASK (GETMONITOR (HUNDRED[X], RStatObj)) Minimum;
         max100:=  ASK (GETMONITOR (HUNDRED[X], RStatObj)) Maximum;
         sd100:=  ASK (GETMONITOR (HUNDRED[X], RStatObj)) StdDev;

         OUTPUT(ObjName[X], "    ", ObjUnit[X]);
         OUTPUT("-----------------------------------------------------");
         OUTPUT("CbtPwr  MEAN      MIN        MAX         STD DEV");
         OUTPUT("50  %: ", mean50/60.0, "  ", min50/60.0, "  ", max50/60.0, "  ",
sd50/60.0);
         OUTPUT("75  %: ", mean75/60.0, "  ", min75/60.0, "  ", max75/60.0, "  ",
sd75/60.0);
         OUTPUT("90  %: ", mean90/60.0, "  ", min90/60.0, "  ", max90/60.0, "  ",
sd90/60.0);
         OUTPUT("100 %: ", mean100/60.0, "  ", min100/60.0, "  ", max100/60.0, "  ",
sd100/60.0);
         OUTPUT;
         ASK MyObj[X] TO PrintArrivalTimes;
         OUTPUT;
        END FOR;

  END METHOD;

END OBJECT;

END MODULE.
```

```
MAIN MODULE Thesis;

{as of 23 MAR 95

NOTES: All objects are being created, with the jumpers
adding themselves to the proper aircraft door VERIFIED 28 FEB}

FROM SimMod         IMPORT StartSimulation, SimTime, StopSimulation,
ResetSimTime;
FROM RandMod        IMPORT RandomObj, FetchSeed;
FROM GrpMod         IMPORT QueueObj;
FROM StatMod        IMPORT SREAL, RStatObj;
FROM ResMod         IMPORT ResourceObj;
FROM IOMod          IMPORT StreamObj, FileUseType(Input);
FROM abn     IMPORT ParatrooperObj, AircraftObj, RightDoor, ObjectiveObj,
ReadFrom, repNum, ReportCollect;
FROM GrpMod         IMPORT QueueObj;

VAR Strm       :StreamObj;
  TextLine     :INTEGER;
  Heading, DoF, StartDeltaX, StartDeltaY, WindSpeedDelta, WindDirDelta,
  PPIX, PPIY, LR, NextAC  :REAL;
  ACType, Passes, X, Y, Z, J, C, OCount, JCount, LCount, RCount,
  Chalks, JumpPass, Race, iterations,
  NumObj, NumJumpDoor, Ch    : INTEGER;
  LeftJump, RightJump :ARRAY INTEGER, INTEGER OF STRING;
  ObjName      :ARRAY INTEGER OF STRING;
  ObjUnit      :ARRAY INTEGER OF STRING;
  ObjX         :ARRAY INTEGER OF REAL;
  ObjY         :ARRAY INTEGER OF REAL;
  ObjForce     :ARRAY INTEGER OF INTEGER;
  ObjSpeed     :ARRAY INTEGER OF REAL;
  MyObj        :ARRAY INTEGER OF ObjectiveObj;
  FIFTY        :ARRAY INTEGER OF SREAL;
  SEVENTYFIVE     :ARRAY INTEGER OF SREAL;
  NINETY       :ARRAY INTEGER OF SREAL;
  HUNDRED:ARRAY INTEGER OF SREAL;
  ACSpeed, Interval, Altitude, DoorInterval          :REAL;
  Rand, Rand2, ACRand       :RandomObj;
  LeftDoorObj :ARRAY INTEGER OF AircraftObj;
  RightDoorObj:ARRAY INTEGER OF RightDoor;
  Jumper, JUMPER :ParatrooperObj;
  unit,YN      :STRING;
  SendEm       :ReadFrom;
  ReportObj    :ReportCollect;
```

```
BEGIN        {Starts off with reading all initialization parameters from
    the requisite .txt files}

    NEW(Strm);
    ASK Strm TO Open("info.txt",Input);
    ASK Strm TO ReadReal(Heading);
    ASK Strm TO ReadReal(PPIX);
    ASK Strm TO ReadReal(PPIY);
    ASK Strm TO ReadInt(Chalks);
    ASK Strm TO ReadInt(ACType);
    ASK Strm TO ReadInt(Passes);
    ASK Strm TO ReadInt(JumpPass);
    ASK Strm TO ReadInt(Race);
    ASK Strm TO ReadInt(NumObj);
    ASK Strm TO ReadInt(NumJumpDoor);
    ASK Strm TO Close;
    DISPOSE(Strm);

    NEW(Strm);{Reads the text files and inputs into 2 dimensional array}
    NEW(LeftJump, 1..NumJumpDoor, 1..Chalks);
    ASK Strm TO Open("Left.txt", Input);
    FOR Y:=1 TO Chalks   {verified 8 FEB}
        FOR X:= 1 TO NumJumpDoor
            ASK Strm TO ReadLine(LeftJump[X, Y]);
        END FOR;
    END FOR;
    ASK Strm TO Close;
    DISPOSE(Strm);

    NEW(Strm);{ditto}
    NEW(RightJump, 1..NumJumpDoor, 1..Chalks);
    ASK Strm TO Open("Right.txt", Input);
    FOR Y:=1 TO Chalks   {verified 8 FEB}
        FOR X:= 1 TO NumJumpDoor
            ASK Strm TO ReadLine(RightJump[X, Y]);
        END FOR;
    END FOR;
    ASK Strm TO Close;
    DISPOSE(Strm);


    {reads and establishes points for Objectives}
    NEW(ObjName, 1..NumObj);
    NEW(ObjUnit, 1..NumObj);
NEW(ObjX, 1..NumObj);
NEW(ObjY, 1..NumObj);
NEW(ObjForce, 1..NumObj);
NEW(ObjSpeed, 1..NumObj);

NEW(Strm);
ASK Strm TO Open("ObjName.txt", Input);
FOR X:= 1 TO NumObj
        ASK Strm TO ReadLine(ObjName[X]);
END FOR;
ASK Strm TO Close;
DISPOSE(Strm);

NEW(Strm);
ASK Strm TO Open("ObjUnit.txt", Input);
FOR X:= 1 TO NumObj
        ASK Strm TO ReadLine(ObjUnit[X]);
END FOR;
ASK Strm TO Close;
DISPOSE(Strm);

NEW(Strm);
ASK Strm TO Open("ObjX.txt", Input);
FOR X:= 1 TO NumObj
        ASK Strm TO ReadReal(ObjX[X]);
END FOR;
ASK Strm TO Close;
DISPOSE(Strm);

NEW(Strm);
ASK Strm TO Open("ObjY.txt", Input);
FOR X:= 1 TO NumObj
        ASK Strm TO ReadReal(ObjY[X]);
END FOR;
ASK Strm TO Close;
DISPOSE(Strm);

NEW(Strm);
ASK Strm TO Open("ObjForce.txt", Input);
FOR X:= 1 TO NumObj
        ASK Strm TO ReadInt(ObjForce[X]);
END FOR;
ASK Strm TO Close;
DISPOSE(Strm);
```

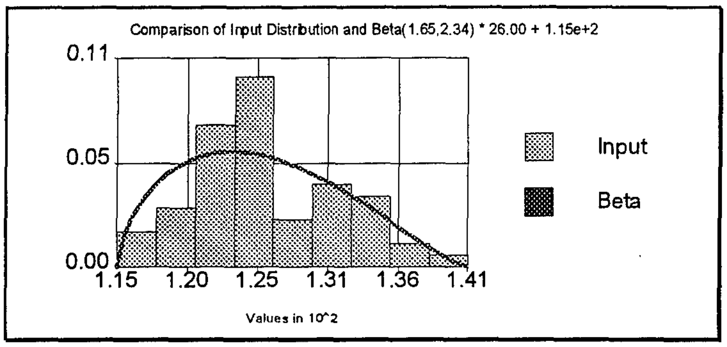
```
    NEW(Strm);
    ASK Strm TO Open("ObjSpeed.txt", Input);
    FOR X:= 1 TO NumObj
        ASK Strm TO ReadReal(ObjSpeed[X]);
    END FOR;
    ASK Strm TO Close;
    DISPOSE(Strm);

{********** END Entrance Parameter Input **********************}

    OUTPUT("Beginning MODSIM Mass Tactical Airborne Simulation ..... ");
    OUTPUT;


    {Generate the AC}
    NEW(LeftDoorObj,1..Chalks); {allocate the space}
    NEW(RightDoorObj,1..Chalks);
   FOR X := 1 TO Chalks
        NEW(RightDoorObj[X]);
        NEW(LeftDoorObj[X]);
    END FOR;

    NEW(Rand);
    {calculates the observed variances in the serial}
    ACSpeed:= ASK Rand Gamma(125.75, 496.0);  {VALID}
    DoF:= Heading + ASK Rand Triangular(-10.0, 0.0, 10.0);
    WindSpeedDelta:=ASK Rand Normal(5.0, 2.0);
    WindDirDelta:=ASK Rand UniformReal(0.0, 360.0);
    DISPOSE(Rand);

    {Make all Main Variables Global and accessible to all modules}
    NEW(SendEm);
    ASK SendEm TO InitGlobalINT(Passes, Chalks, JumpPass, Race, NumObj,
NumJumpDoor);
    ASK SendEm TO InitGlobalREAL(DoF, PPIX, PPIY, ACSpeed, Interval,
WindSpeedDelta, WindDirDelta);
    ASK SendEm TO InitArrays;
    FOR X:= 1 TO NumObj
     ASK SendEm TO InitArrayString(ObjName[X], ObjUnit[X]);
     ASK SendEm TO InitArrayReal(ObjX[X], ObjY[X], ObjSpeed[X]);
     ASK SendEm TO InitArrayInt(ObjForce[X]);
    END FOR;
    ASK SendEm TO ReadManifest;
    ASK SendEm TO InitObjectives;
    ASK SendEm TO PrintGlobals;
    {Activate the report collector}
    NEW(ReportObj);


FOR iterations := 1 TO 7  {Initially 10 reps}

    INC(repNum);
    ResetSimTime(0.0);  {resets for each replication}
    {Conduct aircraft position initialization}
    NEW(Rand2);
    ASK Rand2 TO SetSeed(repNum);
    FOR Ch:= 1 TO Chalks
        StartDeltaX:=ASK Rand2 Triangular(-240.0, 0.0, 240.0);
        StartDeltaY:=ASK Rand2 Triangular(-240.0, 0.0, 240.0);
        LR:= ASK Rand2 Normal(1.54381, 0.259065);
        ASK LeftDoorObj[Ch] TO StartUp(StartDeltaX, StartDeltaY);
        ASK RightDoorObj[Ch] TO StartUpR(StartDeltaX, StartDeltaY, LR);
    END FOR;


    {GREENLIGHT!!!}

    FOR C:= 1 TO Chalks
        TELL LeftDoorObj[C] TO DropEmL(C);
        TELL RightDoorObj[C] TO DropEmR(C);
    END FOR;


    ACSpeed:= ASK Rand2 Gamma(125.75, 496.0);   {Update for next pass}
    DoF:= Heading + ASK Rand2 Triangular(-10.0, 0.0, 10.0);
    WindSpeedDelta:=ASK Rand2 Normal(5.0, 2.0);
    WindDirDelta:=ASK Rand2 UniformReal(0.0, 360.0);
    ASK SendEm TO UpDateNextPass(DoF, ACSpeed, WindSpeedDelta,
WindDirDelta);
    {NEXT PASS !!!!!!}
    StartSimulation;
    ASK ReportObj TO RepReport(repNum);  {gives replication stats}
    {StopSimulation;}
END FOR;

    ASK ReportObj TO FinalReport;

    OUTPUT;
    OUTPUT("In order to get a hard copy of the above results,");
```

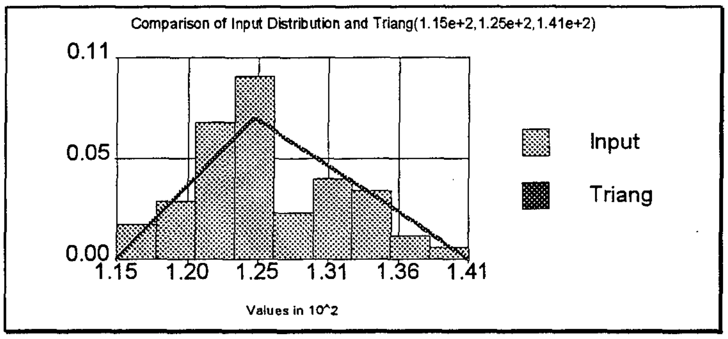
```
        OUTPUT("Select File above, and place a check next to");
        OUTPUT("'stdout.txt' to send a copy of the output");
        OUTPUT("to a file called  c:\manifest\stdout.txt");
        OUTPUT("  or you can highlight a desired section and copy/cut/paste");
        OUTPUT("directly to a word processor or the notepad...");
        OUTPUT;
        OUTPUT("Enter any character and press return when complete...");
        INPUT(Z);

END MODULE.
```

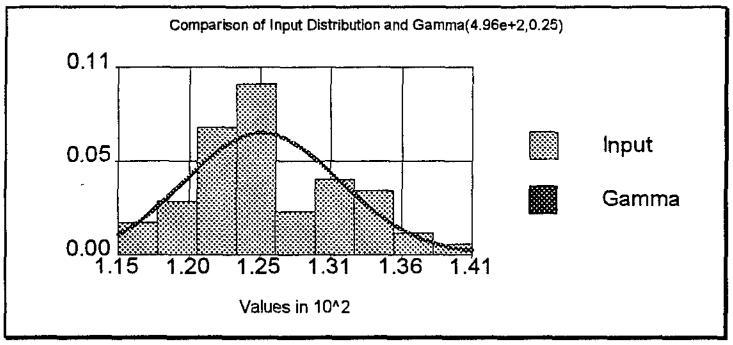
APPENDIX C

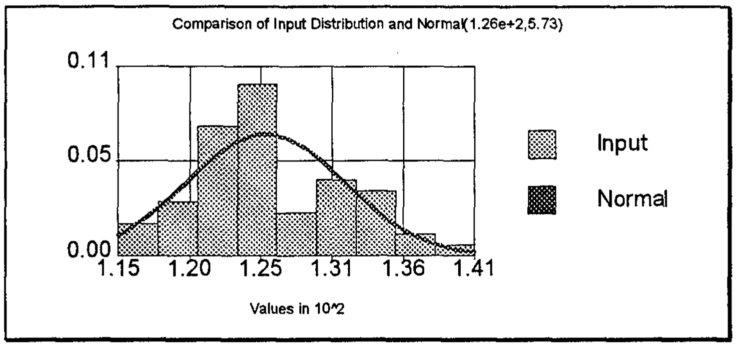
SIMULATION DISTRIBUTION DATA

Aircraft Speed (Knots per Hour)
Observed vs. Selected Distributions





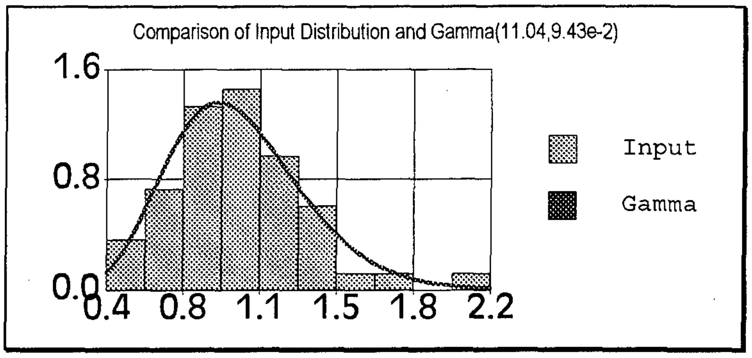


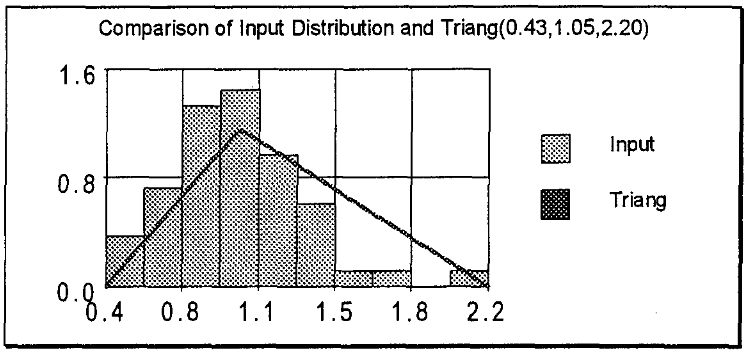


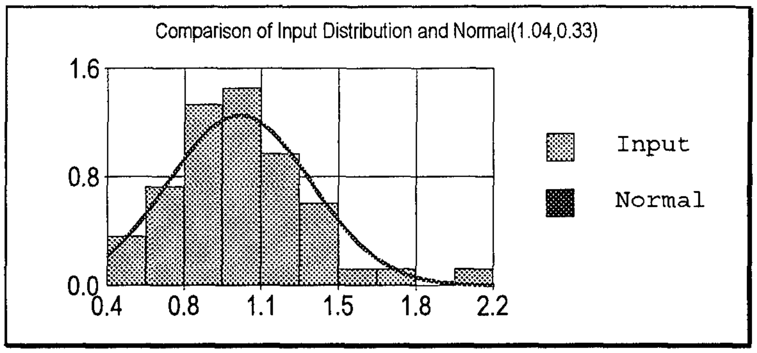
Aircraft Speed in Knots

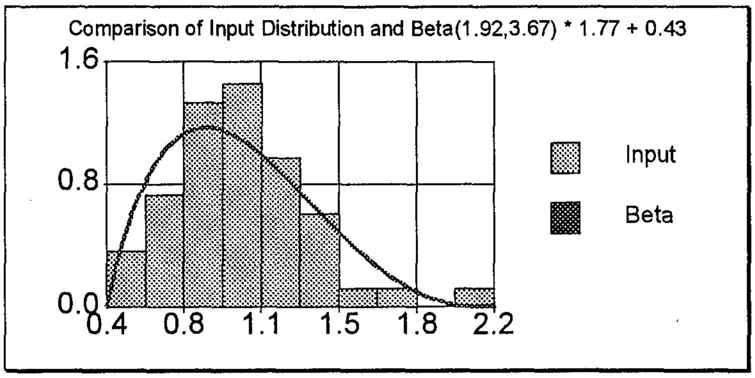
	Input Data	Triang(1.15e+2,1.25e+2,1.41e+2)	Gamma(4.96e+2,0.25)	Normal(1.26e+2,5.73)	Beta(1.65,2.34) * 26.00 + 1.15e+2
Minimum=	115				
Maximum=	141				
Mode=	125.111111	125.111111	125.496328	125.75	0.32678
Mean=	125.75	127.037037	125.75	125.75	0.413462
Std Deviation=	5.727922	9.655521	5.647937	5.727922	0.220305
Variance=	32.809091	93.229081	31.899191	32.809091	0.048534
Skewness=	0.404434	0.210335	0.089828	0	0.262019
Kurtosis=	2.721867	2.38793	3.012104	3	2.682333
Input Settings:					
Type of Fit:	MLEs Only				
Tests Run:	Chi-Square		K-S Test		A-D Test
Histogram:					
Min:	115	115	115	115	115
Max:	141	141	141	141	141
P1:	0.018544	0.010989	0.018284	0.018612	0.029907
P2:	0.030907	0.032967	0.03815	0.037188	0.051763
P3:	0.074176	0.054945	0.059562	0.057616	0.059597
P4:	0.098901	0.076923	0.07053	0.069217	0.059299
P5:	0.024725	0.062937	0.064149	0.064477	0.053348
P6:	0.043269	0.048951	0.045346	0.046573	0.043378
P7:	0.037088	0.034965	0.025188	0.026084	0.030773
P8:	0.012363	0.020979	0.011108	0.011328	0.017002
P9:	6.18E-03	6.99E-03	3.93E-03	3.81E-03	4.21E-03
# Classes=	9				
Best Fit Results					
C-S Test		7.434416	7.726335	8.007671	9.960928
C-S Rank		1	2	3	4
K-S Test		0.188734	0.121177	0.127911	0.118781
K-S Rank		4	2	3	1
A-D Test		2.493095	0.532974	0.580823	0.758282
A-D Rank		4	1	2	3

Time Between Jumper Exits on Single Door (in seconds)
Observed vs. Selected Distributions


Comparison of Input Distribution and Gamma(11.04,9.43e-2)


Comparison of Input Distribution and Triang(0.43,1.05,2.20)

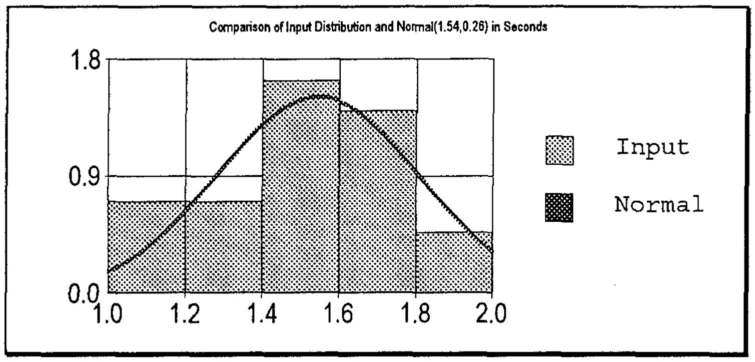

Comparison of Input Distribution and Normal(1.04,0.33)

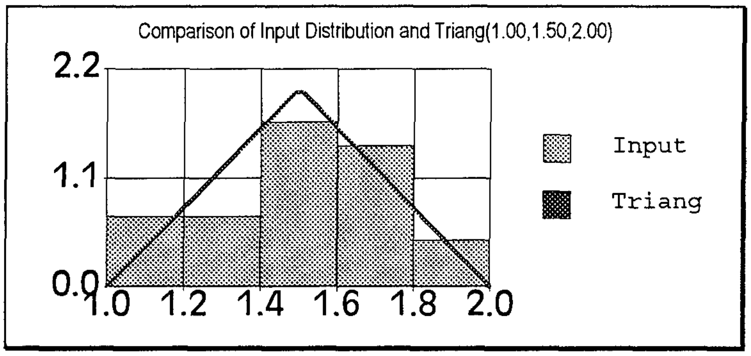

Comparison of Input Distribution and Beta(1.92,3.67) * 1.77 + 0.43

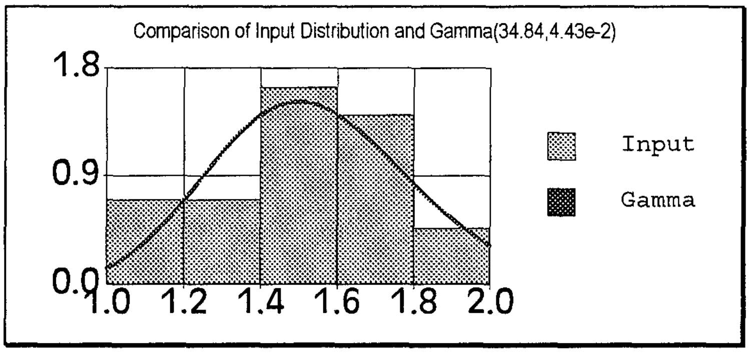
Jumper Exit Interval in Seconds

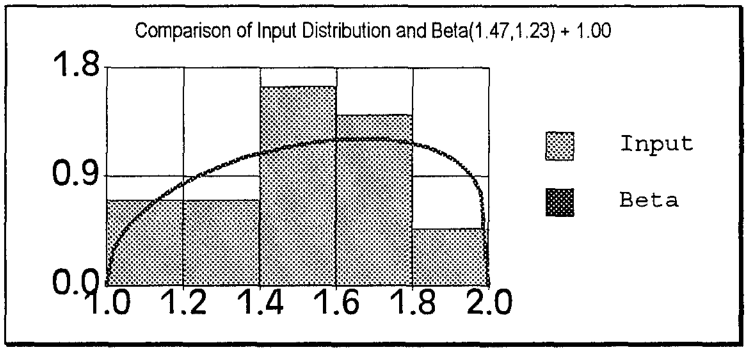
	Input Data	Gamma(11.04,9.43e-2)	Triang(0.43,1.05,2.20)	Normal(1.04,0.33)	Beta(1.92,3.67) * 1.77 + 0.43	Expon(1.04)
Minimum=	0.433					
Maximum=	2.2					
Mode=	1.05145	0.946632	1.05145	1.040958	0.257147	0
Mean=	1.040958	1.040958	1.22815	1.040958	0.344062	1.040958
Std Deviation=	0.32704	0.313353	0.385844	0.32704	0.185082	1.040958
Variance=	0.106955	0.09819	0.148875	0.106955	0.034255	1.083594
Skewness=	1.013351	0.602048	0.27709	0	0.444124	2
Kurtosis=	4.752971	3.543692	2.387842	3	3.201565	9
Input Settings:						
Type of Fit:	MLEs Only					
Tests Run:	Chi-Square		K-S Test		A-D Test	
Histogram:						
Min:	0.433	0.433	0.433	0.433	0.433	0.433
Max:	2.2	2.2	2.2	2.2	2.2	2.2
P1:	0.353707	0.302136	0.161695	0.345265	0.483479	0.582178
P2:	0.707414	0.868443	0.485084	0.704009	0.990519	0.491288
P3:	1.296925	1.284272	0.808473	1.072068	1.136768	0.414587
P4:	1.414827	1.25026	1.131862	1.219229	1.058881	0.349861
P5:	0.943218	0.913021	0.957729	1.03554	0.855399	0.29524
P6:	0.589511	0.540499	0.783597	0.656852	0.602948	0.249146
P7:	0.117902	0.272651	0.609464	0.311162	0.359993	0.210249
P8:	0.117902	0.121247	0.435332	0.110084	0.167543	0.177424
P9:	0	0.048696	0.261199	0.029086	0.048188	0.149725
P10:	0.117902	0.017981	0.087066	5.74E-03	2.86E-03	0.126349
# Classes=	10					
Best Fit Results						
C-S Test		6.427143	13.94462	20.657597	43.491597	62.782943
C-S Rank		1	2	3	4	5
K-S Test		0.095932	0.295571	0.134707	0.117516	0.399145
K-S Rank		1	4	3	2	5
A-D Test		0.306562	7.473672	0.733605	0.806843	11.251995
A-D Rank		1	4	2	3	5

Time Between Left and Right Door First Jumper Exit (in seconds)
Observed vs. Selected Distributions





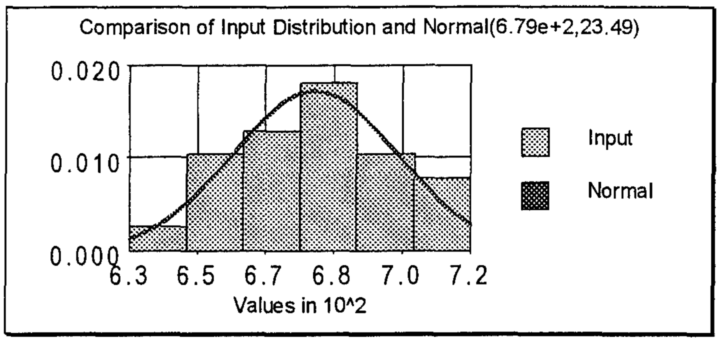


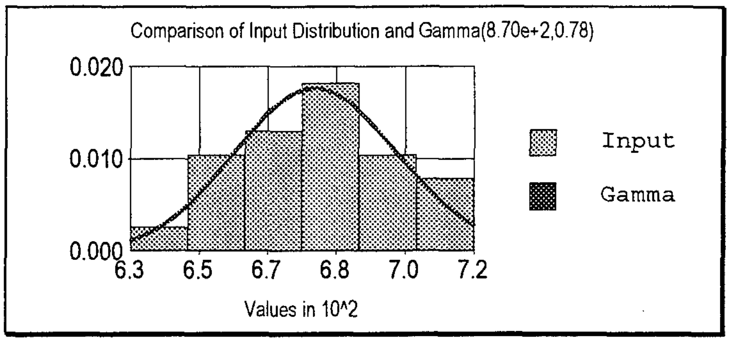


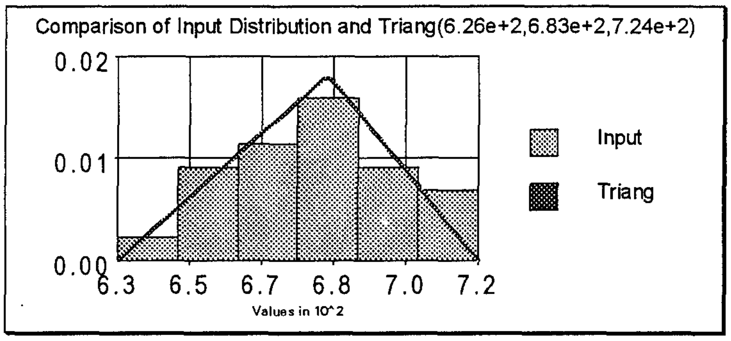
Left and Right Door First Jumper Exit Interval in Seconds

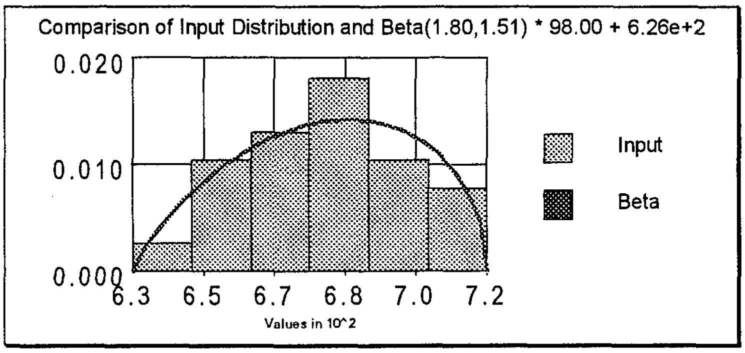
	Input Data	Normal(1.54,0.26)	Triang(1.00,1.50,2.00)	Gamma(34.84,4.43e-2)	Beta(1.47,1.23) + 1.00	Expon(1.54)
Minimum=	1					
Maximum=	2					
Mode=	1.5	1.54381	1.5	1.499501	0.669633	0
Mean=	1.54381	1.54381	1.5	1.54381	0.54381	1.54381
Std Deviation=	0.259065	0.259065	0.263523	0.261542	0.259065	1.54381
Variance=	0.067115	0.067115	0.069444	0.068404	0.067115	2.383348
Skewness=	-0.299	0	-1.68E-15	0.338827	-0.144032	2
Kurtosis=	2.314516	3	2.38805	3.172206	1.647319	9
Input Settings:						
Type of Fit:	MLEs Only					
Tests Run:	Chi-Square		K-S Test		A-D Test	
Histogram:						
Min:	1	1	1	1	1	1
Max:	2	2	2	2	2	2
P1:	0.714286	0.354995	0.4	0.355437	0.636955	0.317659
P2:	0.714286	0.988952	1.2	1.111007	1.003426	0.279061
P3:	1.666667	1.518068	2	1.543907	1.178468	0.245152
P4:	1.428571	1.284016	1.2	1.169198	1.225804	0.215364
P5:	0.47619	0.598429	0.4	0.552472	1.070413	0.189195
# Classes=	5					
Best Fit Results						
C-S Test		2.08199	2.34	2.443512	2.765124	70.082625
C-S Rank		1	2	3	4	5
K-S Test		0.107665	0.200952	0.117981	0.128203	0.476777
K-S Rank		1	4	2	3	5
A-D Test		0.197126	1.719698	0.299161	0.90623	6.77273
A-D Rank		1	4	2	3	5

Jumper-Under-Canopy Altitude Above Ground (in Feet)
Observed vs. Selected Distributions


Comparison of Input Distribution and Normal(6.79e+2,23.49)


Comparison of Input Distribution and Gamma(8.70e+2,0.78)


Comparison of Input Distribution and Triang(6.26e+2,6.83e+2,7.24e+2)


Comparison of Input Distribution and Beta(1.80,1.51) * 98.00 + 6.26e+2

Jumper Altitude Under Canopy in Feet

	Input Data	Normal(6.79e+2,23.49)	Beta(1.80,1.51) * 98.00 + 6.26e+2	Gamma(8.70e+2,0.78)	Triang(6.26e+2,6.83e+2,7.24e+2)
Minimum=	626				
Maximum=	724				
Mode=	683.166667	679.291667	0.610315	678.511309	683.166667
Mean=	679.291667	679.291667	0.543793	679.291667	677.722222
Std Deviation=	23.493716	23.493716	0.239732	23.023697	48.908093
Variance=	551.95471	551.95471	0.057471	530.090645	2392.001543
Skewness=	-0.030139	0	-0.137435	0.067787	-0.159833
Kurtosis=	2.552438	3	1.764313	3.006893	2.387981
Input Settings:					
Type of Fit:	MLEs Only				
Tests Run:	Chi-Square		K-S Test		A-D Test
Histogram:					
Min:	626	626	626	626	626
Max:	724	724	724	724	724
P1:	2.55E-03	2.68E-03	4.38E-03	2.49E-03	2.92E-03
P2:	0.010204	8.01E-03	9.54E-03	8.09E-03	8.75E-03
P3:	0.012755	0.014754	0.012647	0.015218	0.014577
P4:	0.017857	0.016751	0.013945	0.016986	0.020408
P5:	0.010204	0.01173	0.013131	0.011532	0.012245
P6:	7.65E-03	5.07E-03	8.78E-03	4.87E-03	4.08E-03
# Classes=	6				
Best Fit Results					
C-S Test		0.967895	1.059248	1.072375	1.685714
C-S Rank		1	2	3	4
K-S Test		0.089319	0.114914	0.092	0.089015
K-S Rank		2	4	3	1
A-D Test		0.192314	0.975439	0.194971	1.122398
A-D Rank		1	3	2	4

	Jumper Lateral Drift Calculations					
Throw	***DriftAlt***	***(100')***	***Truncated***	***Delta***		
78	724	7.24	7	0.24		
130	672	6.72	6	0.72		
81	721	7.21	7	0.21		
115	687	6.87	6	0.87		
125	677	6.77	6	0.77		
143	659	6.59	6	0.59		
90	712	7.12	7	0.12		
176	626	6.26	6	0.26		
147	655	6.55	6	0.55		
106	696	6.96	6	0.96		
112	690	6.9	6	0.9		
120	682	6.82	6	0.82		
153	649	6.49	6	0.49		
108	694	6.94	6	0.94		
125	677	6.77	6	0.77		
119	683	6.83	6	0.83		
107	695	6.95	6	0.95		
128	674	6.74	6	0.74		
144	658	6.58	6	0.58		
141	661	6.61	6	0.61		
144	658	6.58	6	0.58		
139	663	6.63	6	0.63		
100	702	7.02	7	0.02		
114	688	6.88	6	0.88		
	679.2917	6.792917		28.518	drift in yards	
	Drift Ft.	26.08806		0.9144		
				26.07686	drift in meters	
			Inferenced --> Triangular (-26,0,+26)			

APPENDIX D

NET SOLVE AND EXCEL SOLUTIONS ANALYTICAL MODEL VERIFICATION

NETSOLVE -- VERSION 1.3.1 -- JUNE 1991

```
>LIST NODES
 NAME           SUPPLY
 ----          --------
 1                5.00
 10               5.00
 11               5.00
 12               5.00
 13               5.00
 14               5.00
 15               5.00
 16               5.00
 17               5.00
 18               5.00
 19               5.00
 2                5.00
 20               5.00
 3                5.00
 4                5.00
 5                5.00
 6                5.00
 7                5.00
 8                5.00
 9                5.00
 A              -31.00
 B              -29.00
 C              -26.00

>LIST EDGES
 FROM    TO              COST       LOWER      UPPER
 ----    --          --------    --------   --------
 1       A             806.00        0.00   999999.00
 1       B             658.00        0.00   999999.00
 1       C             721.00        0.00   999999.00
 10      A             874.00        0.00   999999.00
 10      B            1996.00        0.00   999999.00
 10      C            1116.00        0.00   999999.00
 11      A            1000.00        0.00   999999.00
 11      B            2145.00        0.00   999999.00
 11      C            1245.00        0.00   999999.00
 12      A            1133.00        0.00   999999.00
 12      B            2295.00        0.00   999999.00
 12      C            1379.00        0.00   999999.00
 13      A            1269.00        0.00   999999.00
 13      B            2445.00        0.00   999999.00
 13      C            1515.00        0.00   999999.00
 14      A            1408.00        0.00   999999.00
 14      B            2595.00        0.00   999999.00
 14      C            1654.00        0.00   999999.00
 15      A            1550.00        0.00   999999.00
 15      B            2744.00        0.00   999999.00
 15      C            1795.00        0.00   999999.00
 16      A            1692.00        0.00   999999.00
 16      B            2894.00        0.00   999999.00
 16      C            1937.00        0.00   999999.00
 17      A            1836.00        0.00   999999.00
 17      B            3044.00        0.00   999999.00
```

```
17        C            2081.00          0.00     999999.00
18        A            1981.00          0.00     999999.00
18        B            3194.00          0.00     999999.00
18        C            2225.00          0.00     999999.00
19        A            2126.00          0.00     999999.00
19        B            3344.00          0.00     999999.00
19        C            2370.00          0.00     999999.00
2         A             695.00          0.00     999999.00
2         B             804.00          0.00     999999.00
2         C             649.00          0.00     999999.00
20        A            2272.00          0.00     999999.00
20        B            3493.00          0.00     999999.00
20        C            2515.00          0.00     999999.00
3         A             600.00          0.00     999999.00
3         B             952.00          0.00     999999.00
3         C             605.00          0.00     999999.00
4         A             532.00          0.00     999999.00
4         B            1101.00          0.00     999999.00
4         C             597.00          0.00     999999.00
5         A             500.00          0.00     999999.00
5         B            1249.00          0.00     999999.00
5         C             626.00          0.00     999999.00
6         A             513.00          0.00     999999.00
6         B            1398.00          0.00     999999.00
6         C             688.00          0.00     999999.00
7         A             566.00          0.00     999999.00
7         B            1547.00          0.00     999999.00
7         C             774.00          0.00     999999.00
8         A             650.00          0.00     999999.00
8         B            1697.00          0.00     999999.00
8         C             877.00          0.00     999999.00
9         A             755.00          0.00     999999.00
9         B            1846.00          0.00     999999.00
9         C             992.00          0.00     999999.00

>TRANS
   TRANSPORTATION PROBLEM:    MINIMUM COST IS     101540.00

     FROM      TO             LOWER        FLOW       UPPER          COST
     ----      --          --------    --------    --------      --------
     11        A               0.00        5.00   999999.00       1000.00
     12        A               0.00        5.00   999999.00       1133.00
     13        A               0.00        5.00   999999.00       1269.00
     14        A               0.00        5.00   999999.00       1408.00
     15        A               0.00        1.00   999999.00       1550.00
     16        A               0.00        5.00   999999.00       1692.00
     17        A               0.00        5.00   999999.00       1836.00
     1         B               0.00        5.00   999999.00        658.00
     2         B               0.00        5.00   999999.00        804.00
     3         B               0.00        5.00   999999.00        952.00
     4         B               0.00        5.00   999999.00       1101.00
     5         B               0.00        5.00   999999.00       1249.00
     6         B               0.00        4.00   999999.00       1398.00
     10        C               0.00        5.00   999999.00       1116.00
     15        C               0.00        4.00   999999.00       1795.00
     18        C               0.00        1.00   999999.00       2225.00
     6         C               0.00        1.00   999999.00        688.00
     7         C               0.00        5.00   999999.00        774.00
```

```
 8        C                0.00         5.00   999999.00       877.00
 9        C                0.00         5.00   999999.00       992.00

SENSITIVITY ANALYSIS FOR EDGE COSTS

 NODE                DUAL
 ----                ----
 1                2567.00
 10               1399.00
 11               1270.00
 12               1137.00
 13               1001.00
 14                862.00
 15                720.00
 16                578.00
 17                434.00
 18                290.00
 19                290.00
 2                2421.00
 20                  0.00
 3                2273.00
 4                2124.00
 5                1976.00
 6                1827.00
 7                1741.00
 8                1638.00
 9                1523.00
 A                2270.00
 B                3225.00
 C                2515.00

                 EDGE      REDUCED                  COST RANGE
FROM     TO      STATE        COST        LOWER      CURRENT       UPPER
----     --      -----     -------        -----      -------       -----
1        A       LOWER     1103.00      -297.00       806.00   999999.00
1        B       BASIC        0.00   -999999.00       658.00     1431.00
1        C       LOWER      773.00       -52.00       721.00   999999.00
10       A       LOWER        3.00       871.00       874.00   999999.00
10       B       LOWER      170.00      1826.00      1996.00   999999.00
10       C       BASIC        0.00   -999999.00      1116.00     1119.00
11       A       BASIC        0.00   -999999.00      1000.00     1000.00
11       B       LOWER      190.00      1955.00      2145.00   999999.00
11       C       LOWER        0.00      1245.00      1245.00   999999.00
12       A       BASIC        0.00   -999999.00      1133.00     1134.00
12       B       LOWER      207.00      2088.00      2295.00   999999.00
12       C       LOWER        1.00      1378.00      1379.00   999999.00
13       A       BASIC        0.00   -999999.00      1269.00     1270.00
13       B       LOWER      221.00      2224.00      2445.00   999999.00
13       C       LOWER        1.00      1514.00      1515.00   999999.00
14       A       BASIC        0.00   -999999.00      1408.00     1409.00
14       B       LOWER      232.00      2363.00      2595.00   999999.00
14       C       LOWER        1.00      1653.00      1654.00   999999.00
15       A       BASIC        0.00      1550.00      1550.00     1551.00
15       B       LOWER      239.00      2505.00      2744.00   999999.00
15       C       BASIC        0.00      1794.00      1795.00     1795.00
16       A       BASIC        0.00   -999999.00      1692.00     1692.00
16       B       LOWER      247.00      2647.00      2894.00   999999.00
16       C       LOWER        0.00      1937.00      1937.00   999999.00
17       A       BASIC        0.00   -999999.00      1836.00     1836.00
17       B       LOWER      253.00      2791.00      3044.00   999999.00
17       C       LOWER        0.00      2081.00      2081.00   999999.00
18       A       LOWER        1.00      1980.00      1981.00   999999.00
```

```
18        B        LOWER       259.00      2935.00     3194.00   999999.00
18        C        BASIC         0.00   -999999.00     2225.00     2226.00
19        A        LOWER       146.00      1980.00     2126.00   999999.00
19        B        LOWER       409.00      2935.00     3344.00   999999.00
19        C        LOWER       145.00      2225.00     2370.00   999999.00
2         A        LOWER       846.00      -151.00      695.00   999999.00
2         B        BASIC         0.00   -999999.00      804.00     1359.00
2         C        LOWER       555.00        94.00      649.00   999999.00
20        A        LOWER         2.00      2270.00     2272.00   999999.00
20        B        LOWER       268.00      3225.00     3493.00   999999.00
20        C        BASIC         0.00   -999999.00     2515.00     2517.00
3         A        LOWER       603.00        -3.00      600.00   999999.00
3         B        BASIC         0.00   -999999.00      952.00     1315.00
3         C        LOWER       363.00       242.00      605.00   999999.00
4         A        LOWER       386.00       146.00      532.00   999999.00
4         B        BASIC         0.00   -999999.00     1101.00     1307.00
4         C        LOWER       206.00       391.00      597.00   999999.00
5         A        LOWER       206.00       294.00      500.00   999999.00
5         B        BASIC         0.00   -999999.00     1249.00     1336.00
5         C        LOWER        87.00       539.00      626.00   999999.00
6         A        LOWER        70.00       443.00      513.00   999999.00
6         B        BASIC         0.00      1311.00     1398.00     1461.00
6         C        BASIC         0.00       625.00      688.00      758.00
7         A        LOWER        37.00       529.00      566.00   999999.00
7         B        LOWER        63.00      1484.00     1547.00   999999.00
7         C        BASIC         0.00   -999999.00      774.00      811.00
8         A        LOWER        18.00       632.00      650.00   999999.00
8         B        LOWER       110.00      1587.00     1697.00   999999.00
8         C        BASIC         0.00   -999999.00      877.00      895.00
9         A        LOWER         8.00       747.00      755.00   999999.00
9         B        LOWER       144.00      1702.00     1846.00   999999.00
9         C        BASIC         0.00   -999999.00      992.00     1000.00
```

posn	A	B	C	Null		A	B	C	Null				costs:			
1	806	658	721	0		0	5	0	0	5	5		0	3290	-6.4E-13	0
2	695	804	649	0		0	5	0	0	5	5		0	4020	-6.4E-29	0
3	600	952	605	0		0	5	0	0	5	5		0	4760	0	0
4	532	1101	597	0		0	5	0	0	5	5		0	5505	0	0
5	500	1249	626	0		0	5	0	0	5	5		0	6245	0	0
6	513	1398	688	0		0	4	1	0	5	5		0	5592	688	0
7	566	1547	774	0		0	0	5	0	5	5		0	0	3870	0
8	650	1697	877	0		0	0	5	0	5	5		0	0	4385	0
9	755	1846	992	0		0	0	5	0	5	5		-6.7E-13	0	4960	0
10	874	1996	1116	0		0	0	5	0	5	5		0	0	5580	0
11	1000	2145	1245	0		1	0	4	0	5	5		1000	0	4980	0
12	1133	2295	1379	0		5	0	0	0	5	5		5665	0	0	0
13	1269	2445	1515	0		5	0	0	0	5	5		6345	0	0	0
14	1408	2595	1654	0		5	0	0	0	5	5		7040	0	0	0
15	1550	2744	1795	0		5	0	0	0	5	5		7750	0	0	0
16	1692	2894	1937	0		5	0	0	0	5	5		8460	0	0	0
17	1836	3044	2081	0		5	0	0	0	5	5		9180	0	1.38E-19	0
18	1981	3194	2225	0		0	0	1	4	5	5		0	0	2225	0
19	2126	3344	2370	0		0	0	0	5	5	5		0	0	0	0
20	2272	3493	2515	0		0	0	0	5	5	5		0	0	0	0
						31	29	26	14							
					reqs->	31	29	26	14							
											Total Transportation Cost->			101540		

APPENDIX E

DETERMINATION OF NUMBER OF SIMULATION RUNS

In order to acquire a requisite degree of accuracy from the simulation model, the number of replication runs per scenario had to be determined. The rule of thumb used was that the standard error of the mean ($\approx StdDev \div \sqrt{\# Re\,plications}$), should be within 10 percent of the simulation mean. In order to test that the selected number of replications was sufficient, a 2 factor, Fisher experimental design was developed.

Experiment

The factors and their levels are explained below.

1. Number of Objectives (3-Low, 8-High)
2. Relative Distance from Line of Flight to the Objectives (Low <500 meters, High >1000 meters)

Four experimental runs were designated as being,

1. 3 Objectives, Close Together
2. 3 Objectives, Far Apart
3. 8 Objectives, Close Together
4. 8 Objectives, Far Apart

Each experimental run was iterated through the simulation model, and the number of runs varied until all combat power percentages from 90% and lower achieved the desired results with *standard error of the mean* less than 10% of the mean. The one hundred percent rate was not used because it is there that any outliers would reside.

Results

It was found that after just seven replications, that all parameters fell within the required 10% of the mean.

The following pages contain a spreadsheet roll-up of the comparison data, as well as the simulation run summary tables.

EXPERIMENT 1		(3 Objs, Close Together)				
		mean	std dev	10% mean	Std Error Mean	Difference
1	50%	20.299	1.98	2.0299	0.748369657	-1.28153
	75%	29.248	3.38	2.9248	1.277519919	-1.64728
	90%	35.638	3.64	3.5638	1.375790682	-2.18801
	100%	66.791	18.71	6.6791	7.07171529	***0.392615***
2	50%	25.353	2.58	2.5353	0.97514834	-1.56015
	75%	33.686	3.18	3.3686	1.201927024	-2.16667
	90%	42.038	3.32	4.2038	1.25484205	-2.94896
	100%	103.441	25.15	10.3441	9.505806496	-0.83829
3	50%	23.577	2.92	2.3577	1.103656261	-1.25404
	75%	31.371	3.58	3.1371	1.353112813	-1.78399
	90%	41.020	3.75	4.102	1.417366774	-2.68463
	100%	87.698	17.00	8.7698	6.425396041	-2.3444
EXPERIMENT 2		(3 Objs, Far Apart)				
		mean	std dev	10% mean	Std Error Mean	Difference
1	50%	40.210	2.68	4.021	1.012944788	-3.00806
	75%	61.560	3.91	6.156	1.477841089	-4.67816
	90%	70.140	3.31	7.014	1.251062406	-5.76294
	100%	144.498	44.49	14.4498	16.8156394	***2.365839***
2	50%	52.718	4.12	5.2718	1.557213629	-3.71459
	75%	69.433	5.77	6.9433	2.180855009	-4.76244
	90%	87.640	7.08	8.764	2.674098647	-6.0899
	100%	195.980	32.73	19.598	12.36888738	-7.22911
3	50%	35.876	3.51	3.5876	1.3266553	-2.26094
	75%	48.695	4.65	4.8695	1.757534799	-3.11197
	90%	57.298	6.33	5.7298	2.390625292	-3.33917
	100%	134.510	30.95	13.451	11.69724451	-1.75376

*(**Bold Italics** Signify exceeds 10%)*

EXPERIMENT 3		*(8 Objs, Close Together)*				
		mean	std dev	10% mean	Std Error Mean	Difference
1	50%	15.223	1.81	1.5223	0.684115696	-0.83818
	75%	20.927	4.43	2.0927	1.674382615	-0.41832
	90%	24.900	4.40	2.49	1.661909788	-0.82809
	100%	28.468	3.95	2.8468	1.492959668	-1.35384
2	50%	6.380	1.16	0.638	0.438438789	-0.19956
	75%	8.690	1.44	0.869	0.542379019	-0.32662
	90%	11.415	1.56	1.1415	0.588490684	-0.55301
	100%	14.487	1.40	1.4487	0.528394333	-0.92031
3	50%	16.151	2.92	1.6151	1.103656261	-0.51144
	75%	21.130	3.22	2.113	1.218179497	-0.89482
	90%	28.950	6.37	2.895	2.406121835	-0.48888
	100%	50.828	25.65	5.0828	9.694032804	***4.611233***
4	50%	7.529	1.20	0.7529	0.455069226	-0.29783
	75%	10.470	1.74	1.047	0.656146325	-0.39085
	90%	15.554	3.39	1.5554	1.282546846	-0.27285
	100%	24.725	11.58	2.4725	4.376450633	***1.903951***
5	50%	8.527	1.90	0.8527	0.71737657	-0.13532
	75%	11.790	2.67	1.179	1.00803125	-0.17097
	90%	16.712	4.05	1.6712	1.530000187	-0.1412
	100%	25.602	10.45	2.5602	3.950862636	***1.390663***
6	50%	11.916	3.29	1.1916	1.243125152	0.051525
	75%	15.228	3.46	1.5228	1.308513006	-0.21429
	90%	20.370	3.81	2.037	1.4415565	-0.59544
	100%	28.112	3.79	2.8112	1.432107388	-1.37909
7	50%	13.371	2.07	1.3371	0.780496637	-0.5566
	75%	18.106	2.74	1.8106	1.034110798	-0.77649
	90%	26.369	3.51	2.6369	1.327033265	-1.30987
	100%	42.641	11.75	4.2641	4.440326629	***0.176227***
8	50%	19.446	1.52	1.9446	0.575261928	-1.36934
	75%	24.283	1.19	2.4283	0.450533652	-1.97777
	90%	34.046	4.55	3.4046	1.720872246	-1.68373
	100%	65.850	21.37	6.585	8.077856717	***1.492857***

*(**Bold Italics** Signify exceeds 10%)*

		mean	std dev	10% mean	Std Error Mean	Difference
1	50%	15.223	1.81	1.5223	0.684115696	-0.83818
	75%	20.927	4.43	2.0927	1.674382615	-0.41832
	90%	24.900	4.40	2.49	1.661909788	-0.82809
	100%	28.468	3.95	2.8468	1.492959668	-1.35384
2	50%	6.380	1.16	0.638	0.438438789	-0.19956
	75%	8.690	1.44	0.869	0.542379019	-0.32662
	90%	11.415	1.56	1.1415	0.588490684	-0.55301
	100%	14.487	1.40	1.4487	0.528394333	-0.92031
3	50%	16.151	2.92	1.6151	1.103656261	-0.51144
	75%	21.130	3.22	2.113	1.218179497	-0.89482
	90%	28.950	6.37	2.895	2.406121835	-0.48888
	100%	50.828	25.65	5.0828	9.694032804	***4.61123***
4	50%	7.529	1.20	0.7529	0.455069226	-0.29783
	75%	10.470	1.74	1.047	0.656146325	-0.39085
	90%	15.554	3.39	1.5554	1.282546846	-0.27285
	100%	24.725	11.58	2.4725	4.376450633	***1.90395***
5	50%	8.527	1.90	0.8527	0.71737657	-0.13532
	75%	11.790	2.67	1.179	1.00803125	-0.17097
	90%	16.712	4.05	1.6712	1.530000187	-0.1412
	100%	25.602	10.45	2.5602	3.950862636	***1.39066***
6	50%	11.916	3.29	1.1916	1.243125152	0.051525
	75%	15.228	3.46	1.5228	1.308513006	-0.21429
	90%	20.370	3.81	2.037	1.4415565	-0.59544
	100%	28.112	3.79	2.8112	1.432107388	-1.37909
7	50%	13.371	2.07	1.3371	0.780496637	-0.5566
	75%	18.106	2.74	1.8106	1.034110798	-0.77649
	90%	26.369	3.51	2.6369	1.327033265	-1.30987
	100%	42.641	11.75	4.2641	4.440326629	***0.17623***
8	50%	19.446	1.52	1.9446	0.575261928	-1.36934
	75%	24.283	1.19	2.4283	0.450533652	-1.97777
	90%	34.046	4.55	3.4046	1.720872246	-1.68373
	100%	65.850	21.37	6.585	8.077856717	***1.49286***

*(**Bold Italics** Signify exceeds 10%)*

experiment run #1
three objectives, close together

```
----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
----------------------------------------
      (All Statistics in minutes)

"OBJ ONE"    "AAA"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 20.299183  17.563965  23.297879  1.983177
75   %: 29.248033  24.121691  34.262161  3.384608
90   %: 35.638321  31.330072  42.191993  3.646340
100 %: 66.791143  45.646914  102.461469  18.715339

"OBJ TWO"    "BBB"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 25.353038  22.682515  30.309705  2.580599
75   %: 33.686861  30.114739  39.037380  3.184983
90   %: 42.038775  36.622607  46.809389  3.322664
100 %: 103.441486  82.537711  153.174786  25.135060

"OBJ THREE"    "CCC"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 23.577606  18.194399  26.219224  2.923911
75   %: 31.371785  25.887346  35.332681  3.581196
90   %: 41.020454  35.554298  45.998704  3.754633
100 %: 87.698490  50.367149  107.928840  17.001104
```

experiment run #2
three objectives, far apart

```
----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
----------------------------------------
      (All Statistics in minutes)

"OBJ ONE"    "AAA"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 40.218070  37.506394  44.682011  2.685986
75   %: 61.544415  55.109418  66.124016  3.913103
90   %: 70.144510  66.104679  76.196656  3.307076
100 %: 144.498574  94.590266  212.851862  44.492560
```

```
"OBJ TWO"    "BBB"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 52.718545  48.398425  58.961249  4.128783
75   %: 69.433122  63.385890  81.167963  5.775218
90   %: 87.640244  80.696023  99.466124  7.075481
100 %: 195.981259  142.703197  238.420424  32.725932

"OBJ THREE"   "CCC"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 35.875864  29.948320  40.933414  3.510423
75   %: 48.694676  39.263221  54.436277  4.655325
90   %: 57.298803  45.112577  67.398113  6.325739
100 %: 134.510260  87.038972  171.173906  30.948834
```

experiment run #3
eight objectives, close together

```
-----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
-----------------------------------------
      (All Statistics in minutes)

"OBJ ONE"    "AAA"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 15.223775  12.158681  17.632476  1.814161
75   %: 20.927171  14.770610  27.033183  4.433034
90   %: 24.901000  19.399812  30.866300  4.397522
100 %: 28.468146  23.068756  34.629598  3.954194

"OBJ TWO"    "BBB"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 6.386609  4.288305  8.013062  1.156419
75   %: 8.692678  6.833973  11.477626  1.435960
90   %: 11.415266  8.311717  13.158111  1.557283
100 %: 14.487829  11.939651  16.702840  1.398893

"OBJ THREE"   "CCC"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 16.151042  12.652192  20.827993  2.901987
75   %: 21.130546  16.762091  25.821051  3.223696
90   %: 28.950406  20.935689  37.839606  6.366919
100 %: 50.828219  22.460834  99.286355  25.648783
```

```
"OBJ FOUR"   "DDD"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 7.529433  5.871306  9.047550  1.201224
75   %: 10.474001  8.061830  13.407667  1.736637
90   %: 15.554296  11.801359  21.281656  3.393352
100 %: 24.725152  15.990805  51.007608  11.579073

"OBJ FIVE"   "EEE"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 8.527428  6.130111  12.139355  1.898984
75   %: 11.797495  8.778640  17.184188  2.667364
90   %: 16.712477  10.414687  21.705893  4.048730
100 %: 25.602238  14.132604  42.876734  10.453538

"OBJ SIX"   "FFF"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 11.916645  9.261768  17.262338  3.284402
75   %: 15.228856  11.879630  20.876659  3.462195
90   %: 20.379718  16.428834  27.238586  3.814633
100 %: 28.112873  22.625618  33.647948  3.789684

"OBJ SEVEN"   "GGG"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 13.370435  10.028754  15.055701  2.065499
75   %: 18.106706  14.836333  23.144314  2.736766
90   %: 26.369231  19.670571  29.073175  3.508981
100 %: 42.641170  30.244348  67.781304  11.748696

"OBJ EIGHT"   "HHH"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 19.446705  17.132884  21.718480  1.522292
75   %: 24.283712  22.629760  26.262374  1.192376
90   %: 34.046928  28.900672  42.998962  4.553989
100 %: 65.850399  38.757492  100.178069  21.372503
```

experiment run #4
eight objectives, far apart

```
----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
----------------------------------------
      (All Statistics in minutes)

"OBJ ONE"    "AAA"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX           STD DEV
50   %: 32.802135  28.324900  37.595003  2.780911
75   %: 46.672803  35.908702  58.096387  8.412596
90   %: 55.361943  47.963138  64.550912  5.482680
100 %: 60.202796  53.988963  67.962869  5.646248

"OBJ TWO"    "BBB"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX           STD DEV
50   %: 48.147732  40.837361  57.296749  5.448646
75   %: 63.220517  56.601906  70.952540  4.930249
90   %: 81.503128  71.469409  94.194432  7.616252
100 %: 129.150222  72.382758  228.939093  52.596934

"OBJ THREE"    "CCC"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX           STD DEV
50   %: 26.547709  23.928680  31.116420  2.460109
75   %: 35.709220  27.075498  42.690487  5.581574
90   %: 44.110891  36.175440  51.038894  4.259537
100 %: 49.372585  42.678869  56.750381  4.015753

"OBJ FOUR"    "DDD"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX           STD DEV
50   %: 28.887835  22.630624  38.696752  5.098526
75   %: 38.749143  29.957251  44.090861  4.692637
90   %: 48.084083  37.258670  69.484634  9.596734
100 %: 71.157467  38.694891  153.813965  36.531628

"OBJ FIVE"    "EEE"
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX           STD DEV
50   %: 47.367622  44.674345  52.520944  2.627210
75   %: 60.884863  57.010442  67.913748  3.415511
90   %: 74.839564  69.738619  83.952084  4.983582
100 %: 99.381919  73.271258  143.107550  21.281400
```

```
"OBJ SIX"   "FFF"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50  %: 38.036085  31.235435  49.549798  6.113140
75  %: 49.135257  44.091707  55.313266  4.383204
90  %: 62.875188  52.834356  76.887946  7.195424
100 %: 92.214859  60.928166  189.657111  42.533834

"OBJ SEVEN"   "GGG"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50  %: 45.410632  41.475602  48.446228  2.050644
75  %: 58.094243  52.904528  63.190122  3.668135
90  %: 74.594274  65.237435  88.782497  7.230258
100 %: 119.161357  68.408433  190.663211  45.486253

"OBJ EIGHT"   "HHH"
----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50  %: 52.946910  46.472994  58.591117  3.488680
75  %: 72.845813  61.163892  84.776236  6.990516
90  %: 93.407914  82.047960  109.061772  7.754103
100 %: 137.291634  105.484729  234.666018  40.647613
```

```
29 Mar 95 SPSS for MS WINDOWS Release 6.1
Page 1

Data written to the working file.
4 variables and 154 cases written.
Variable: OBJ        Type: Number   Format: F11.2
Variable: DISTANCE   Type: Number   Format: F11.2
Variable: RESPONSE   Type: Number   Format: F11.2
Variable: OBJECTIV   Type: Number   Format: F9.3

            * * * A N A L Y S I S   O F   V A R I A N C E * * *

                 OBJECTIV objective
            by   DISTANCE
                 OBJ

                 UNIQUE sums of squares
                 All effects entered simultaneously


                                   Sum of                 Mean
Sig
Source of Variation               Squares     DF        Square          F
of F

Main Effects                    10380.341      2      5190.171    167.324
.000
   DISTANCE                      8934.668      1      8934.668    288.042
.000
   OBJ                           1445.674      1      1445.674     46.607
.000

2-Way Interactions                  3.758      1         3.758       .121
.728
   DISTANCE OBJ                     3.758      1         3.758       .121
.728

Explained                       12921.789      3      4307.263    138.860
.000

Residual                         4652.797    150        31.019

Total                           17574.586    153       114.867

154 cases were processed.
0 cases (.0 pct) were missing.
```

APPENDIX F

SIMULATION RUNS AND HYPOTHESIS TESTING

VALIDATION OF INITIAL ASSUMPTIONS

In order to validate the original assumptions made by the analytical model, the simulation model was used as a platform to test their validity. Three test were conducted and compared to a baseline simulation run. All jumper arrival times were collected and compared against the baseline for each test. The statistical method utilized to determine if the induced changes had a significant effect was a paired-*t* test. The paired t test was selected because it is more robust than the z test because independence between the data streams does not have to be assumed, nor does the variances have to be assumed equal. The formulas used for the mean, variance and confidence interval are as follows:

mean $$\overline{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}$$ *with Z being the delta between the baseline and the test*

n = number of the sample
j = the counter

variance $$\hat{Var}[\overline{Z}(n)] = \frac{\sum_{j=1}^{n}[Z_j - \overline{Z}(n)]^2}{n(n-1)}$$

confidence interval $$\overline{Z}(n) \pm t_{n-1,1-\alpha/2}\sqrt{\hat{Var}[\overline{Z}(n)]}$$

The three runs were conducted using a 90% confidence interval (alpha = 0.10), and the test results showed that if zero remained in the interval, that the variations were relatively insignificant. If zero is not in the interval, then the effect was significant.

Shifting of the PPI Location by more than 150 meters laterally :
(equivalent to unforcasted 5 knot cross-wind)

0.066851	-0.58368	Overall
0.063693	-0.56225	Dagger
0.011676	-1.0174	Knife
0.140409	-0.18078	Sword

Turning on Green Light 2 Seconds Late (equivalent to 150 meters)

0.365751	-0.00461	Overall
0.111061	-0.34774	Dagger
0.944598	0.353082	Knife
0.131759	-0.11228	Sword

Shift in Direction of Flight by 5 degrees

0.149303	-0.23161	Overall

The initial baseline simulation summary tables, as well as the three experiments are included below:

```
BASELINE DATA FOR PARAMETER COMPARISONS

-----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
-----------------------------------------
      (All Statistics in minutes)

DAGGER   Tm Assault
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 16.613037  13.032731  19.784089  1.946222
75   %: 19.922472  14.874712  24.092071  2.545627
90   %: 23.930910  16.958924  28.331914  3.695188
100  %: 31.510434  19.818276  47.742606  8.473805


KNIFE   Tm Secure
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 16.201435  14.073770  18.682875  1.749613
75   %: 19.451268  15.663541  23.570423  2.943065
90   %: 22.998035  17.251743  26.645862  3.271955
100  %: 31.201911  26.028330  35.556528  3.201599
```

```
SWORD   Tm Support
-------------------------------------------------------
CbtPwr  MEAN       MIN         MAX          STD DEV
50   %: 15.838115  13.319718  18.149162  1.696988
75   %: 18.063340  14.787694  21.956413  2.263400
90   %: 21.165166  17.989184  24.450185  2.606980
100 %: 32.997018  20.390402  50.754493  10.959171


CHANGE IN DIRECTION OF FLIGHT
by 5 degrees


----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
----------------------------------------
      (All Statistics in minutes)

DAGGER   Tm Assault
-------------------------------------------------------
CbtPwr  MEAN       MIN         MAX          STD DEV
50   %: 17.030831  13.304015  20.514785  2.085511
75   %: 20.449224  15.083081  25.139837  2.794134
90   %: 24.689810  17.514951  29.627069  3.786888
100 %: 32.101909  20.111357  49.575960  8.863965

KNIFE   Tm Secure
-------------------------------------------------------
CbtPwr  MEAN       MIN         MAX          STD DEV
50   %: 17.227929  14.844812  19.797962  1.878559
75   %: 20.807543  17.102138  25.598799  3.076524
90   %: 24.498609  18.409583  28.901120  3.311140
100 %: 33.964872  29.378827  41.442003  3.829729

SWORD   Tm Support
-------------------------------------------------------
CbtPwr  MEAN       MIN         MAX          STD DEV
50   %: 14.545943  12.613244  17.261585  1.769113
75   %: 16.845795  14.734352  21.089179  2.025344
90   %: 19.306523  15.937919  23.577724  2.688693
100 %: 28.412991  17.271131  45.888063  9.639580
```

```
LATE GREEN LIGHT - 2 Seconds
150 meters on the ground

-----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
-----------------------------------------
      (All Statistics in minutes)

DAGGER   Tm Assault
------------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 16.907175  13.837262  20.344117  2.091180
75   %: 19.705693  15.342740  24.058521  2.527990
90   %: 24.141380  17.287803  29.766695  3.972132
100 %: 30.769142  18.981200  48.568852  8.925679


KNIFE   Tm Secure
------------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 15.644027  13.167811  18.456912  1.750206
75   %: 18.426222  15.350288  23.222247  2.604989
90   %: 21.775327  16.139604  26.113457  3.066720
100 %: 29.362203  25.221552  35.666572  3.472683


SWORD   Tm Support
------------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 15.580000  12.679087  17.436743  1.641355
75   %: 17.805949  14.092808  20.497389  2.039780
90   %: 20.235041  15.642598  23.652487  2.518216
100 %: 32.560633  21.435744  49.243243  10.468685


120 METER ALTERATION IN PPI
5 knot unforcasted crosswind

-----------------------------------------
<<<<<  SIMULATION SUMMARY REPORT  >>>>>>>
-----------------------------------------
      (All Statistics in minutes)

DAGGER   Tm Assault
------------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50   %: 17.011799  13.986819  19.978353  1.695954
75   %: 20.753806  15.278547  24.280021  2.617668
90   %: 25.052353  18.211756  29.531282  3.401935
100 %: 34.262483  21.997361  50.507182  8.566131
```

```
KNIFE   Tm Secure
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50  %: 17.134291  14.386569  20.355092  2.049114
75  %: 20.935112  16.185310  26.815133  3.499616
90  %: 25.432129  18.817393  29.685246  3.684252
100 %: 34.821895  26.483244  39.747294  4.032667


SWORD   Tm Support
-----------------------------------------------------
CbtPwr  MEAN       MIN        MAX          STD DEV
50  %: 16.261273  14.000717  18.939151  1.871534
75  %: 18.802796  16.042364  23.376504  2.314660
90  %: 22.086457  16.784713  25.222171  3.114058
100 %: 34.122766  20.814483  55.344388  11.407534
```

Analytical Solver Manifest : Both Left and Right Door					
Jumper Position	Chalk Number One	Two	Three	Four	Five
1	Tm Support	Tm Support	Tm Support	Tm Support	Tm Support
2	Tm Support	Tm Support	Tm Support	Tm Support	Tm Support
3	Tm Support	Tm Support	Tm Support	Tm Support	Tm Support
4	Tm Support	Tm Support	Tm Support	Tm Support	Tm Support
5	Tm Support	Tm Support	Tm Support	Tm Support	Tm Support
6	Tm Support	Tm Support	Tm Support	Tm Support	Tm Support
7	Tm Support	Tm Support	Tm Support	Tm Support	Tm A/2-325
8	Tm Support	Tm Support	Tm Support	Tm Support	Tm A/2-325
9	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
10	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
11	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
12	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
13	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
14	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
15	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
16	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
17	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
18	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325	Tm A/2-325
19	Tm A/2-326	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
20	Tm A/2-327	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
21	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
22	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
23	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
24	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
25	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
26	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
27	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
28	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325	Tm B/2-325
29	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
30	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
31	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
32	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
33	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
34	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
35	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
36	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
37	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
38	Tm Breach	Tm Breach	Tm Breach	Tm Breach	Tm Breach
39	Tm Breach	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
40	Tm Breach	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
41	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
42	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
43	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
44	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
45	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
46	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
47	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
48	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
49	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
50	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325	Tm C/2-325
51	""""	""""	""""	""""	""""
52	""""	""""	""""	""""	""""
53	D/2-325	D/2-325	D/2-325	D/2-325	""""
54	D/2-325	D/2-325	D/2-325	D/2-325	""""
55	D/2-325	D/2-325	D/2-325	D/2-325	D/2-325
56	D/2-325	D/2-325	D/2-325	D/2-325	D/2-325
57	D/2-325	D/2-325	D/2-325	D/2-325	D/2-325
58	D/2-325	D/2-325	D/2-325	D/2-325	D/2-325
59	D/2-325	D/2-325	D/2-325	D/2-325	D/2-325
60	D/2-325	D/2-325	D/2-325	D/2-325	D/2-325

Manually Derived Manifest : Both Left and Right Door					
Jumper Position	Chalk Number One	Two	Three	Four	Five
1	""""	""""	""""	""""	""""
2	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
3	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
4	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
5	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
6	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
7	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
8	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**
9	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**
10	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**
11	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
12	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
13	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
14	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**	**Tm A/2-325**
15	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**
16	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm A/2-325**
17	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm A/2-325**
18	**Tm A/2-326**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
19	**Tm A/2-327**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
20	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
21	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
22	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
23	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
24	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
25	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**
26	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**	**Tm Support**
27	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
28	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
29	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**	**Tm B/2-325**
30	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
31	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
32	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
33	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
34	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
35	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
36	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
37	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
38	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
39	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
40	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**	**Tm C/2-325**
41	**Tm C/2-325**	**Tm Breach**	**Tm Breach**	**Tm C/2-325**	**Tm C/2-325**
42	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
43	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
44	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
45	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
46	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
47	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
48	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
49	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
50	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
51	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**	**Tm Breach**
52	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**
53	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**
54	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**
55	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**
56	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**
57	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**
58	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	""""
59	**D/2-325**	**D/2-325**	**D/2-325**	**D/2-325**	""""
60	""""	""""	""""	""""	""""

	means	Var	90% Confidence Interval		Conclusions:		Points
OBJ HEPI	10.16	0.404	11.21	9.115	Analaytic Superior		2.3
OBJ TOWER	-2.19	0.091	-1.7	-2.69	Manual Superior		5.2
OBJ SNOW	-0.13	0.099	0.391	-0.64	Not Conclusive		1
OBJ FALCON	-4.84	2.571	-2.2	-7.48	Manual Superior		1
OBJ GREEN	12.8	0.179	13.49	12.1	Analytic Superior		2.3
OBJ BLOCK	0.801	0.183	1.505	0.097	Analytic Superior		2.3
n=	253						
					Qualitative		
					Analytic	6.9	
					Manual	6.2	